Universidad Carlos III de Madrid

www.uc3m.es

Escuela Politécnica Superior

Grado en Ingeniería de Sistemas Audiovisuales

TRABAJO DE FIN DE GRADO

Aesthetics Assessment of Videos through Visual Descriptors and Automatic Polarity Annotation

Alejandro Hernández García

Tutor: Fernando Fernández Martínez

4th March 2014

Agradecimientos

Aunque no soy muy partidario de este tipo de agradecimientos para encabezar un documento académico, ahora que escribo las últimas líneas de un proyecto que se ha llevado gran parte de mi esfuerzo de un año y que representa el cierre de otros cuatro años de carrera, siento la necesidad y me veo en la obligación de acordarme de las personas que han contribuído a alcanzar esta meta.

En primer lugar, no puedo sino agradecer la dedicación de mi tutor Fernando, con quien he compartido este camino. Ha sabido guiar este trabajo de investigación por la buena senda, corregirme cuando ha sido necesario, enseñarme y, lo que más valoro, reconocer con sinceridad el esfuerzo, que ha sido, sin duda, el motor para llegar hasta aquí.

También me acuerdo en estos momentos de mis padres, a quienes debo la educación y los valores que me han inculcado y de los que hoy puedo sentirme orgulloso. También de mis hermanos y el resto de mi pequeña gran familia, que igualmente han contribuído a que hoy sea la persona que soy.

Y partícipes de la educación y la formación que hoy tengo han sido todos los profesores que he tenido, desde primaria hasta la universidad, pasando por el bachillerato. Si bien considero que todas las personas tienen algo que enseñarnos, creo que quienes dedican con vocación su tiempo a nuestra educación se merecen un reconocimiento especial. En particular, a profesores como Aida, María, Ana o Jerónimo los recordaré siempre.

Tampoco puedo olvidarme de mis amigos, que han sido siempre mi vía de escape de las rutinas y el estrés y con quienes he pasado muchos de mis momentos más felices. Gracias a mis amigos de la universidad por llenar de alegría mi día a día de la carrera, a todos mis compañeros del Erasmus en Stuttgart por hacer de aquel uno de los mejores años de mi vida y a mis amigos de Villalba por estar a mi lado siempre, año tras año desde que éramos unos enanos. Y, de manera especial, gracias María por compartirlo todo conmigo, por soportarme en los peores momentos y hacerme vivir los mejores.

23 de febrero de 2014

Resumen en español

En un mundo en el que las nuevas tecnologías están cada vez más ligadas a la información multimedia, el desarrollo de herramientas que permitan manejar fácilmente este tipo de datos se ha convertido en una tarea imprescindible, que ha despertado el interés científico en los últimos años. De entre las líneas de investigación que han empezado a desarrollarse recientemente, el estudio de características subjetivas en material audiovisual a partir de datos objetivos es de especial interés por cuanto puede ser aplicado a sistemas de clasificación y de recomendación. Este documento presenta un trabajo de investigación centrado en el estudio de modelos que permitan predecir automáticamente la satisfacción o interés que despierta un vídeo, concretamente un anuncio publicitario de un coche, en los usuarios de YouTube que lo ven, a partir de los descriptores de bajo nivel del vídeo. Un aspecto novedoso de este trabajo es el planteamiento de una solución para este tipo de problemas basada en un procedimiento para obtener automáticamente el etiquetado de los vídeos mediante técnicas de aprendizaje no supervisado.

Para ello, se ha adquirido un conjunto de anuncios de coches junto con los metadatos asociados a cada vídeo que proporcionan los usuarios y que ofrecen información referente a la satisfacción que perciben estos cuando los visualizan en YouTube. Estos metadatos han permitido diseñar tres estrategias de análisis cluster para anotar automáticamente los vídeos, utilizando cada una de ellas un conjunto de metadatos diferente, de acuerdo a la manera en que los mismos son proporcionados por los usuarios. Por otro lado, se ha extraído, mediante técnicas de procesamiento de imagen y vídeo, un conjunto descriptores visuales de cada vídeo para posteriormente entrenar un sistema de aprendizaje máquina que ha permitido el estudio de la relevancia y utilidad de este conjunto de descriptores para predecir el valor estético de los vídeos percibido por los usuarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing growth of video creation and share, specially over the Internet, and the predictable tendency for the future make the development of techniques and tools to handle videos very necessary. In order to improve the efficiency of searching for videos and to offer users satisfactory results, techniques of video classification [6] and video recommendation [1] have been deeply studied in the past and are still in development. However, most used techniques are based on text, tags or metadata. It has been only in recent years that content-based approaches are being researched. A very challenging and valuable tool for improving searches and user experience would be to develop models that allow recognising the aesthetic quality of videos according to what users expect relying on video content.

Here, our purpose with this work is demonstrating that it is possible to determine if a video has been positively or negatively perceived by users, building a predicting model exclusively based on low-level video descriptors and using as ground truth the labels obtained from *YouTube*[1] metadata inherent to the videos, such as the number of likes or the number of views. These labels will be derived by means of unsupervised learning techniques to later study the correlation between them and the extracted low-level video features classification algorithms.

Hence, this can be seen as trying to build a classification system which, instead of classifying by topic, genre or type of video, is able to determine if a given video is recommendable or not recommendable according to the average and general user perception. The notion of *recommendable* is extremely subjective, so our intention here is to develop a model that allows learning this information through objective data via image and video processing techniques.

1.1 Objectives

Several objectives can be established for this research project. First of all, the main purpose is studying the correlation between a set of low level video features and the sentiment or impression perceived by the viewers of the video. Second, developing a computational model based on unsupervised learning that allows us creating classes that represent the types of videos in terms of the feedback provided by users from the different metadata available at YouTube. Finally, a desirable ultimate goal would be modelling a prediction system which is able to classify a given video as positive or negative with as much accuracy as possible. We can extend the implications of these objectives as follows:

[1] www.youtube.com

- **Study of the correlation of video features with the appeal perceived by users**: in order to accomplish this objective, we first need to have a corpus of videos with their corresponding labels. Then the set of low level video features must be extracted from all the videos and once we have all the information, i.e. the classes to which each video belongs and the values of the video features, we can study to what extent the video descriptors are indicative of the subjective assessment of the videos. Finally, after this study we should be able to determine which features or subsets of features play an important role in the polarity of the perception of the video. With this knowledge, we intend to discover if these features are well-known characteristics that help create attractive or impressing videos or if they are, instead, new features whose importance in the perception of the video was still unknown.
- **Development of a computational model based on unsupervised learning to annotate the corpus**: one of our main contributions is the approach to automatic aesthetics quality assessment through unsupervised learning methods for annotating our corpus. As far as we know, up until now, existing approaches to automatically assess the quality of images and videos have been done by using either already annotated data sets or a user survey to obtain ratings. In this work, we derive the labels for every video by using the freely available metadata provided by YouTube. In order to do so, we make use of techniques of unsupervised learning, such as clustering analysis, and implement three different strategies of combinations of metadata regarding the diverse nature of them.
- **Train and test of a classification system that predicts the user appeal of a video**: Once we achieve the previous objectives, it would be interesting to propose a classification model which predicts, with a certain accuracy, if a new given video will be perceived as positive or negative by users, looking only at its low-level video features. For this purpose, we will perform different experiments with several classification algorithms and configurations.

1.2 Document Structure

This document has been organised with the intention of best reflecting the steps of design, implementation and analysis of results of this research project. Since the development of the research consisted of three different steps (corpus acquisition, feature extraction and classification), the chapters of this document intend to reflect this separation in order to allow the reader understanding the different parts and required processes.

- Chapter 1 is an introduction to the topic in which a brief motivation is offered and the main objectives of the research are outlined.
- Chapter 2 is dedicated to discuss the state of the art in the field of assessing aesthetic quality in multimedia data. Moreover, based on the state of the art, an extended motivation to carry out this research is offered, together with the possible applications that the results may offer.
- Chapter 3 explains in detail all the aspects related to first and necessary step of building a corpus of videos on which the experiments can be performed. This chapter comprises both the operations for acquiring the videos and the process of unsupervised learning that allows deriving the labels to have a completely annotated corpus.
- Chapter 4 is dedicated to the explanation of the implementation of methods and techniques for extracting the video features which will be the base of the experimentation.
- Chapter 5 explains the last step of the implementation, which refers to the supervised learning or classification experiments that will allow extracting results and conclusions.
- The purpose of chapter 6 is to analyse and discuss the results provided by the processes explained in the previous chapters. This chapter is not only a presentation of the results, but useful conclusions and comments are also provided.

- Chapter 7 summarises the main conclusions of this project and suggests some future lines of research.
- Chapter 8 provides some details of the required budget for the project and the schedule of work that has been followed for the execution of this research project.
- Appendix A contains information related to the machine learning software WEKA, which has been used for many tasks in this project.
- Appendix B is a compilation of some of the algorithms, written in pseudo-code, which have been implemented for performing relevant tasks of the project.

Apart from describing the procedures that have been developed to perform the required experiments in this project, each chapter includes sections that introduce the main aspects of tools, techniques or theory that are necessary to correctly understand the operations carried out in this project. This is the case, for instance, of some machine learning techniques that have been used at some steps of this research.

Chapter 2

State of the Art and Proposal

The first aim of this chapter is to provide an overview of the most important scientific works that have been carried out in the fields which are relevant to this research and to give an idea of the current state of research in the area of assessing subjective information from objective data. Once the state of the art is defined, we will provide the motivation for developing this project and a proposal that tries to improve and develop the less explored aspects in our research field.

2.1 Previous Work

This section is a review of the most relevant research works in the study of subjectivity within data by means of computational procedures. We will start with an introduction to recommendation and classification systems, as they are the most important domains of applications of this work, and will follow by exposing the latest works in sentiment analysis, which is a field with important relationships with aesthetics assessment. Finally we will focus on the previous works of aesthetics assessment, both applied on still images and on videos.

2.1.1 Recommendation and Classification Systems

The objectives and applications of this work are closely related to video classification and video recommendation, which are fields of great research interest due to the great amount of available videos today. An important work on video recommendation systems was carried out by Adomavicius and Tuzhilin in 2005 [1], in which they performed a survey of the state of the art at that moment and proposed some improvements. The importance of recommendation systems can be understood by looking at the growth of social networks based on videos and video platforms, such as YouTube. A discussion of the techniques used in the recommendation systems of YouTube is done in [10]. Similarly, video classification techniques have been deeply studied and have still great potential of development. A survey on the literature related to video classification was made in [6] in 2008.

Classification and recommendation systems can be seen as the driving force of other related research works in multimedia applications, such as image and video quality assessment [27, 28], video sentiment analysis or image and video aesthetics assessment.

2.1.2 Sentiment Analysis

The present work aims to extract subjective information from objective data and such a purpose is the goal of a thoroughly researched field such as sentiment analysis or opinion mining [54], which studies the subjectivity of information through automatic computational procedures. Traditionally, sentiment analysis has focused on extracting sentiment and opinions from text sources of different nature [34, 39]. The first attempt to extend sentiment analysis to audiovisual data was recently carried out by Morency *et al.* [33] in 2011, where they perform a multimodal sentiment analysis of 47 videos from YouTube. Together with the text-based sentiment analysis, they take advantage of the extra information that the audiovisual features add. Their conclusion is that using together text, audio (pauses and pitch) and video (smile and look away) improves the performance with respect to using only one kind of feature. Later research following this study has been made in [45, 56].

2.1.3 Aesthetics Assessment

Another field that studies subjectivity is known as aesthetics assessment, which was firstly studied in still images. One of the earliest approaches towards this domain was carried out by Savakis *et al.* [47] more than a decade ago. In that paper, they aimed to find out which aspects were related to image appeal through a ground truth experiment in which 11 participants had to rank 194 pictures belonging to 30 different groups. It was found that image appeal was influenced by image quality only regarding objective aspects, so their conclusion was that image appeal had to be addressed through metrics others than those used for measuring image quality. More recently, in 2006, Datta *et al.* [9] proposed 56 low-level image features tested on 3581 pictures with ratings from the web site *Photo.net* and selected the top 15 features that achieved together an accuracy of 70.12% in separating low from high rated photographs. The features they selected where all based on photographic aspects or well-established rules of thumb, such as brightness, saturation, hue, metrics of usage of the rule of thirds or depth of field indicators. After this successful work, several studies followed this line of research by adding different contributions. This is the case of [22] or [19], where they carry out a higher-level analysis to assess the aesthetic quality of photographs. In 2011, Luca Marchesotti *et al.* [29] extended the study by using a larger and diverse set of features, including generic image descriptors that added statistics computed from low-level local features. Evaluating their models on images collected from *Photo.net* they achieve an accuracy of 89.9%.

However, aesthetics assessment applied to videos has not been addressed until recent years. A related approach was carried out by [35] in 2012, although it was not strictly aesthetic assessment, but a computational model for automatically separating professional videos from amateur ones. Even though the task is not as challenging as modelling a subjective evaluation, they employed an aesthetic approach and achieved 91.2% accuracy. To our knowledge, the first attempt of modelling visual aesthetics in moving images was addressed by Moorthy *et al.* [32] in 2010. They collected 160 consumer videos from YouTube and performed a controlled user study to obtain rating labels as ground truth. Then different frame-level features based on those from [9] and on users reports were computed from the videos and extended to the temporal dimension through a hierarchical pooling method. Finally they selected the most relevant 7 features and after classification procedures they achieved an accuracy of 73.03%. This study was extended by [57], using the same set of videos, but differentiating between semantically independent and dependent features in order to perform a comparative study. Finally, in 2013 [2] uses a lager data set of 1,000 videos and proposes a model which uses features based on psycho-visual statistics.

2.2 Motivation and Proposal

After analysing the previous works on video classification and recommendation, we find that most existing systems use text content such as tags, keywords or data about user interaction for training

the algorithms instead of directly relying on the video content. This fact was probably also observed by the first researchers who developed models to assess aesthetic quality of images and video, since information of aesthetics can be very useful for defining recommendation systems, especially when it is extracted directly from the content. Being able to distinguish between aesthetically appealing pieces from noisy or unappealing content would be highly interesting in platforms, such as YouTube, that currently rely only on textual information. Imagine, for instance, that YouTube was able to classify appealing and unappealing content according to what most users perceive on average. In account of these facts the first successful works in image and video aesthetic assessment were developed [9, 32].

Now, we observe two facts regarding the previous works on video aesthetic assessment. First of all, it is obvious that there is still much work to do and potential improvements that can be done in this field, so this is already an important reason to decide to extend the research. However, there is another important observation to make on the previous work: to the best of our knowledge, up until now, the approaches towards quality assessment in image and video have been faced using as ground truth explicit scores ranked by users. Although this is not a limiting inconvenient, we find very interesting to study the possibility of doing a similar research, but without depending on the ratings of users that have provided their opinion specifically for performing the study.

In this work, we try to approach the problem of video aesthetics assessment without relying on a user survey, but on real information present in YouTube instead. In order to do so, we first collect from YouTube, through automatic queries and filtering, a set of videos together with their metadata, i.e. number of *likes*, *dislikes*, *comments* or *views* among others, to create our corpus. The main idea behind our approach is that we assume these metadata to be indicative of a better or worse appreciation of the video by viewers. For example, it is reasonable to think that a video with many *likes* and a high number of *views* is more appealing from the user point of view than another video with several *dislikes* and a few number of *views*. Under this assumption, we use clustering techniques to bring together videos with similar metadata, create different classes and thus automatically annotate our corpus. Once we have the set of videos with their corresponding labels we carry out well-known image and video processing techniques for extracting low-level features and propose some novel descriptors. Finally we employ different classification algorithms to study how well these features correlate with the user appreciation of the video, taking special notice of how these features can be combined to provide better results.

Thence an important basis for this work is machine learning. Different techniques from this field of artificial intelligence have been used for developing our model. Particularly, the problem we propose can be seen as a classification process with two steps. The first step is an unsupervised learning problem, in which YouTube metadata is used to derive polarity labels aiming to reflect how users have perceived the videos on average. In the second step, these polarity labels are used by supervised learning techniques with the intention of learning, by means of the low-level video features, how the perception of video aesthetics is. Figure 2.1 shows in a diagram the main steps of the project to provide a better overview of the whole process.

At an initial state of the research in automatically assessing the appealing of videos using low-level features, we have preferred not to extend our study to any kind of video because the nature of the huge number of videos available nowadays is extremely diverse and, specially because we use the regular feedback of users in YouTube, we wanted our corpus to be as unbiased as possible. Therefore, we have selected our corpus from a specific domain, where the variations in the content are relatively delimited and thus, the users' feedback in terms of metadata is also as content-independent as possible. Under these important requirements, the domain we have chosen is vehicles advertising videos, where these conditions are reasonably accomplished and we can find an acceptable number of videos and possibly more user feedback than in other domains.

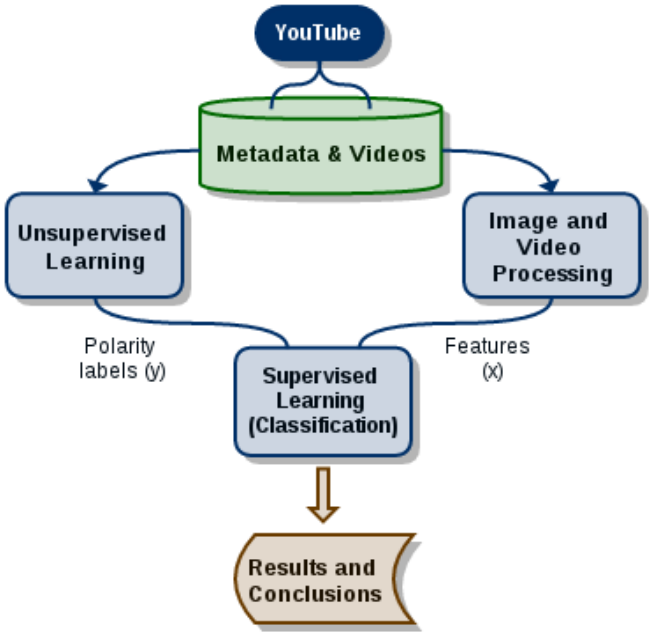


Figure 2.1: Diagram of the project overview

2.3 Applications and Contributions

Since this is a research project, the possible applications of the work will depend strongly on the derived results and should be extended by future work in any case. Nevertheless, we suggest a set of possible applications and derivations of this research work.

First of all, if the objectives are reasonably accomplished, we will obtain some results that will allow us establishing some new knowledge about the importance of the low-level characteristics of a video on the positive or negative impression that it produces on the viewer. Even though we cannot generalise this idea for any kind of video, with this particular work, we should be able to specifically determine which characteristics play an important role in the domain of vehicles advertising. This could be of great interest and application in this particular industry, since it would allow creating spots that potentially enhance the impact on viewers in a positive way.

Furthermore, the creation of a prediction system of the polarity of a video, applied to the particular domain of vehicles advertising, would give the possibility to test a new spot according to the positive or negative impression or feelings that it will create on the public within a certain confidence level.

One important contribution is the approach to aesthetic quality assessment through unsupervised learning techniques by means of YouTube metadata. The fact that these metadata is freely available at YouTube makes it very appropriate and gives this research a direct application and facilitates the future research on the subject without having to perform any user survey to obtain new ratings.

Additionally, the implementation of some new video descriptors could be of interest in the field of image processing for future works and research.

Finally, the most important applications of this work are, on the one hand extending the research on aesthetic quality assessment and sentiment analysis, and on the other hand establishing some bases for future research on these fields, offering new results that will add some knowledge about a relatively unexplored field.

2.4 Requirements

In order to properly develop this research project and to carry out all the necessary operations, the following basic requirements have to be satisfied:

- Computational equipment to develop the algorithms and procedures. This can be satisfied with a personal computer or laptop with a minimum processing capability of around 1.5 GHz and 4 GB of RAM.
- Software. Different computer programmes are required for developing the algorithms and performing other tasks related to the project. Most of them are freely available computer programmes, such as WEKA, Eclipse, Kile or GIMP. Additionally, MATLAB software was established as a requirement, even though a license is necessary.
- Computational equipment specially prepared for running costly tasks, such as video processing applications. For this purpose, the computation cluster of the Departamento de Teoría de la Señal y Comunicaciones of the Universidad Carlos III de Madrid can be used to take advantage of its concurrence and computational capabilities.
- The set of videos that define the corpus on which the experimentation is performed.
- Access to the YouTube API to obtain the videos and the metadata. A developer account is required to download the libraries, sample code and to interact with the YouTube server.
- Documentation. Most research papers can be freely obtained on the Internet, some can be accessed with the credentials of the university and many books are available at the university library.

2.5 Restrictions and Regulatory Framework

In this section, we will discuss the restrictions that we must take into account when developing the tasks involved in the project taking special notice to the terms of service of the YouTube API, which is an important aspect for the videos acquisition process.

2.5.1 Development libraries

Regarding development libraries, most of them can be freely used. This is the case of YouTube API libraries, subject to its terms of service that will be discussed later, Java SE libraries, subject to its license agreement [38], WEKA libraries and the JSOUP HTML parser [17].

MATLAB libraries are not freely available, unless a proper license is owned. In this case, we have used the license owned by the department to be able to use the functionalities provided by MATLAB.

2.5.2 YouTube API Terms of Service

As this research project involves using the YouTube API, it is important to take into account the restrictions imposed by the Terms of Service (ToS) of the API, which can be found in [61]. The two basic principles on which the ToS are based are:

1. Respecting the creators of the content uploaded to YouTube.
2. Predicting all possible situations or use cases that can arise when a developer uses the API and creates applications.

It is explicitly forbidden to sell or distribute parts of the API and allow users of the developed YouTube client to modify the audiovisual content, the users accounts information, to block advertisements, to alter the functionality of YouTube and to exploit copyright-infringing materials. In our case, the developed YouTube Client uses the functionality offered by the API and as a research application does not provide any interaction to third-parties. Furthermore, we have paid special attention to the account information of the authors of the video comments and we have just stored the content of the comments.

Regarding the videos, it was necessary to download them for extracting the video features. For that purpose we have used the software *youtube-dl*, which does not infringe any restriction. Once extracted the features, we did not need to store the videos any more, since we are just interested on the values of the features, not in the content. However, we have to take into account that this is possible as this is a research project. We could not implement an application that exploited economically the videos, as we would infringe copyright restrictions.

Another issue related to copyright concerns the pictures that have been presented in this report in order to illustrate some aspects of the features extractions. Most of these pictures have collected from internet sites which do not apply any restriction. Nevertheless, we have included a reference in each picture indicating the provenance of them.

Chapter 3

Corpus Acquisition

The process of collecting the videos required to perform the experiments has been a long process which consisted of multiple steps and involved different technologies. It is important to note from the beginning that the acquisition process consisted not only of collecting the videos themselves, but, more importantly, of collecting also the metadata inherent to the videos, which is what allows us to define the strategies for deriving the labels that annotate each video in one category or another in terms of the user perception. Furthermore, it is necessary to make a strong filtering in order to be sure of retrieving only videos that fit our requirements.

In this chapter we will first discuss our choice of domain and the requirements that the instances of the corpus must satisfy. Then, we will present the main steps of the process of querying and downloading the videos and their metadata, which has been done through the YouTube API. We will use these requirements to illustrate the filtering step that yields the final configuration of the corpus and, finally, we will describe the procedures for automatically annotating the corpus and the required operations of this process.

3.1 Videos Domain

One of the main aspects of this work in comparison to other related works [32, 57, 2] is that we do not depart from an annotated corpus, but we obtain the labels through unsupervised learning techniques instead. This procedure, as it will be detailed later, consists in deriving or *learning* these labels from the metadata of videos, such as the number of likes or the number of views, which we assume to be indicative of the subjective assessment of the videos by viewers. These metadata are provided by users, as they watch, interact and share videos and therefore we need the content to be as unbiased as possible. For example, if we want to annotate two advertisements in terms of their aesthetic quality and their appeal to viewers by using YouTube metadata, we cannot compare a *Coca-Cola* commercial with a detergent commercial, simply because the diffusion and public interest of the former is, whether we like it or not, much greater and hence, the amount and usability of its metadata will be greater as well. Therefore, in order to minimise this bias, we have restricted our data domain to one single type of videos: car advertisements. The interest and diffusion of this kind of advertisements are bigger compared to other domains and this indicates that the amount of metadata should be reasonably enough.

This is not the only characteristic in favour of the choice of this domain, but the election of advertising videos is also appropriate because the duration is limited, which is important not only for computational reasons, but because the variation of the content is limited as well. We considered using other kind of videos, such as film trailers, but we disregarded this option because

their duration is much longer than commercials' duration. In addition to this, even if there are different car brands advertisements, they all aspire to sell the same product, which plays a key role in the spot, and thus, the polarity differences will depend stronger on the video features than on the content. Nevertheless, despite all these considerations and constraints, we cannot ignore the fact that the content dependency cannot be totally avoided. Finally, publicity is also a desirable domain because of the marketing applications of the research, which could be of interest for many different agents, such as brands, advertising agencies, consumers or public institutions among others.

3.2 Videos Requirements

Having defined our data set domain, car advertisements videos, it is important to remark the requirements we should impose on the retrieved videos to ensure that we can carry out the experiments correctly and with certain confidence. All the particular conditions we will present next are defined to fulfil three general requirements: first of all, we need every video to be a professional car commercial, so that we can ensure a minimum quality. Second, since we intend to derive subjective information from visual features, we need the corpus to be as uniform as possible in order to minimise bias on the features. Finally, but, most importantly, we need every video to have sufficient metadata, i.e. videos must have enough information about the feedback from the users, since we will use these data to automatically derive a polarity label for each video.

Now, we can define the following requirements for a video of the data set:

- It must be a car advertisement.
- It must be a professionally filmed video.
- It should be a recent advertisement, in principle published from 2010 up to 2013. This constraint is established to fulfil the requirement of uniformity in the corpus. Film making has evolved throughout the years and we cannot compare an advertisement from the 90s with a current one in terms of the way they are filmed and produced. To minimise this effect, we define a range of three years for our corpus.
- It should be a Spanish advertisement. The reason for this condition is also corpus uniformity, since advertisements in several countries are sometimes different in order to fit cultural issues, particular to each country or region. Although it is a desirable and generally fulfilled constraint, we have accepted foreign commercials when they were equal or very similar to the ones published in Spain.
- It must have at least three *raters*, i.e. three different users must have clicked the *like* or *dislike* buttons. This strict requirement ensures a minimum feedback from users.

3.3 Collecting Videos and their Metadata

The great amount of video content available today in YouTube, where 100 hours of video are uploaded every minute [64] makes this site a very good option to use as our data provider, even though precisely because of this huge number of videos, filtering procedures are required to deal with this great diversity. Apart from the videos themselves, YouTube provides a number of different metadata inherent to the videos which contain information related to the polarity of users' perception, essential for our purposes of automatically deriving the polarity labels. Some of these metadata are: number of likes, number of dislikes, number of comments and the comments themselves, number of views, rating or number of people who has rated the video. A complete list of all the metadata we extract from the videos is shown in Table 3.1 Because of these characteristics, YouTube was chosen as the source of videos for building the corpus. Although there are other

Video Metadata		
Name		**Example**
videoID		tag:youtube.com,2008:video:H6OfLlJV46Y
URL		http://www.youtube.com/watch?v=H6OfLlJV46Y&feat ...
title		Spot Nuevo Audi A1 Sportback - “Grande en cada detalle”
description		Existen detalles que hacen que las cosas tengan ...
published		2012-03-21T12:52:53.000Z
viewsCount		93094
favoriteCount		0
numComments		32
comments	**user**	user1 (*not shown for privacy reasons*)
	content	La presencia del perro es imponente ...
numRaters		52
rating		4.6153846
numLikes		47
numDislikes		5
aspectRatio		WIDE_SCREEN
duration		33
author		audiespana
locations		España;Andorra
demographicGroups		Hombre, 45-54; Hombre, 35-44; Mujer, 45-54

Table 3.1: Metadata extracted for every video of the corpus. Examples of one video are provided.

sites, such as Vimeo, the amount of videos and the variety are still not comparable to that from YouTube, not to mention the richness of the metadata.

3.3.1 YouTube API

Along all the process of this research project we have tried to get the diverse procedures as automated as possible. Hence, rather than searching and downloading the videos manually from the web page, we turned to the YouTube API, which allows, among many other functions, to raise search queries and retrieve information and metadata related to the videos. Despite being a very useful and essential tool, the YouTube API is still under development and this was an important handicap for the acquisition process. There are three different versions, with *beta* functionalities and continuous updates which make comprehension of the API tricky. Furthermore, the main handicap for developing the programmes was the great lack of documentation regarding the methods and functionalities.

After several tests and attempts, we finally decided to use the YouTube API v2.0. This decision has some advantages and disadvantages. The main advantage is that it was a stable version of the API and this was determining for the decision. On the other hand, YouTube API v3.0 is an improvement over version 2 and therefore it is better adapted to the current state of YouTube, it incorporates more functionalities and the documentation is cleaner. Many functionalities have not been implemented on version 2, and some have been implemented, but not documented, so they remain partially hidden and hard to use. An important aspect of version 2 is that the results are returned as XML tags, while v3.0 uses JSON. The API is available in several programming languages, from which Java was chosen for familiarity reasons, and the programmes have been develop with Eclipse.

The availability of some sample code made it easier to start developing the desired functionality, namely automatically raising a number of queries, retrieve the videos and their related metadata and properly process these results. YouTube API v2.0 is organised in feeds, structured as XML code. For instance, a given query returns a video feed, i.e. an XML document, that contains a

collection of video entries and information for retrieving other related feeds, such as related videos or comment feeds. In turn, each video entry contains the information specific to that video. This information, or, as we call it in this work, metadata, is encapsulated as well in XML tags, most of which are accessible through Java methods, as it is defined in the API documentation [60].

3.3.2 Search Queries

In Section 3.1 we set car advertisements videos as our data domain. In order to obtain as many valid results as possible, we made use of a list of the car brands currently sold in Spain [52]. This list was used to set the car brand name as a search parameter, together with some key words, such as "advertisement", "spot" or "campaign". As stated in the previous section, the retrieval of the videos and their metadata is automatically performed through the YouTube API. In order to optimise the search and to adjust it to the requirements presented in Section 3.2, we established a number of search parameters that YouTube API can interpret, such as the publication date or the language of the results. More specifically, we provided lists including as many lines as car brands, accompanied by search parameters, defined according to the following format:

`<max-num-results>`; `<category-term>`; `<category-label>`; `<car-brand-name>`; `<language>`; `<order-of-results>`; `<min-publication>`; `<max-publication>`; `<keyword1>`, ...,`<keywordN>`

where each parameter has the following meaning:

- `<max-num-results>`: defines the maximum number of results returned by the query. It was set to 60.
- `<category-term>`: defines the YouTube category, in English language, to which the returned videos must belong. We set the categroy "Autos"
- `<category-label>`:defines the YouTube category, in Spanish language, to which the returned videos must belong. We set the category "Motor"
- `<car-brand-name>`: the car brand name for this specific query. It is added to the search as an `AND` parameter, i.e. it is a term that must appear in the results.
- `<language>`: the language of the returned results. As explained in Section 3.2, we prefer Spanish advertisements, so we set it to"es"
- `<order-of-results>`: the order criterion for returning the results. The YouTube API defines four options: *relevance* (default option), *viewsCount*, *rating* and *published*. The results retrieved by changing this parameter are significantly different.
- `<min-publication>`: the minimum publication date of the returned videos. As explained in Section 3.2, the date we set is January, the 1st of 2010.
- `<max-publication>`: the maximum publication date of the returned videos. As explained in Section 3.2, the date we set is January, the 1st of 2013.
- `<keywords>`: the set of different key words that define the search. In order to retrieve advertising videos, we used the words *advertisement*[1], *campaign*[2] and *spot*. These key words are added to search separated by the boolean `OR`, but added all together with an `AND` to the car brand name.

These files are read by the Java programme that executes the queries and the lines are parsed to properly incorporate the parameters to the query. By way of illustration of the effect of these parameters and search terms on the query, one example of query could be the following: "retrieve 60 videos sorted by relevance within the category Autos (or Motor), published after the 1st of January 2010 and before the 1st of January 2013, in Spanish and containing the words [Ford AND (advertisement OR campaign OR spot)]".

[1] anuncio
[2] campaña

For simplicity, we will call *types of queries* the different ways of sorting the results. We make a distinction among these types of queries because the results returned by each type are significantly different in terms of the total number of results and, very importantly, the richness of the metadata of the videos. In total, we built three query files, each for a different type of query: relevance, viewsCount and rating.

3.3.3 Implementation of the Retrieval Algorithm

The algorithm that executes the queries through the YouTube API has been implemented in Java language and developed in Eclipse IDE. The programme that implements the algorithm connects to the YouTube server and is able to generate proper queries and receive video feeds from which to extract the metadata. Most metadata, such as the video identifier, the number of views, the title, the average rating or the duration, among others, are accessible through Java methods defined and documented in the API. However, other metadata, for instance the number of likes and the number of dislikes, are not directly retrievable through documented methods. To overcome this situation, we had to implement specific XML parsers to extract these metadata from the raw XML code. An extreme case is the extraction of information such as the geographical locations from where a video is most watched and the main demographic groups, i.e. gender and age, who watch a video. These data, also known as *insight data* are publicly shown at the YouTube interface for some videos, but they are not accessible through the YouTube API at all. In order to retrieve this potentially useful information, we found out that these statistics are shown at the interface through an AJAX web application, which takes the information from a piece of HTML code embedded into the comments part of an XML file. This XML is the HTTP response of the URL. Therefore, we had to find the URL for requesting this information and use it to build the specific URL for each video. Then we extract the comments of the XML and properly parse the HTML code with JSOUP parser [17] to obtain these hidden statistical data.

The algorithm iterates over the three types of queries presented in the previous section and raises an specific query for each line. Then, it processes the results separately for each type of query, writing two results files as the output for each video, one for the metadata and one for storing all the comments of the video. Algorithm 1 in Appendix B shows a summary of the algorithm operation for raising the queries.

Apart from the basic operations, the implementation of the algorithm deals with a number of problems that may arise during the long execution. Mainly, the most important issue for the process is the *quota* restrictions that YouTube imposes in order to avoid an abuse of the API. Each request to the YouTube server has a cost and there is both a daily limit and and a short-time limit, that avoids performing too many requests in a short period. The daily limit was never exceeded, but the latter became an important obstacle. This situation was handled through three different strategies:

1. Optimising the performance: this was a measure to try to avoid exceeding the restrictions, consisting in minimising the unnecessary requests.

2. Active wait for recovery: this measure was implemented to recover from an error of type “too many recent calls”. It consists in making the programme *sleep* for some time in a stepped manner, trying up to three attempts and restart waiting in case of no successful attempt.

3. Easy restart: this was a last resort strategy: in case the previous method did successfully recover from the quota exception, the current state of the process was saved as a way to restart the process from a given point.

Furthermore, we implemented multiple mechanisms for handling with the possible situations that can arise when working with the YouTube API, such as forbidden videos, forbidden comments and users, metadata with no information, among many others.

3.3.4 Videos Download

Apart from storing all the metadata in files, a main step of the process was the download of the videos. As it has been mentioned, a major concern of this work was to make the procedures as automatic as possible. Therefore, we needed to find a solution for downloading the videos that satisfied this requirement, not just on a whim, but because the amount of videos to download was considerably big.

We first thought of using software such as JDownloader, but we disregarded this option after realising that the usability of this programme from a command line interpreter was quite limited and thus, the programmatic implementation of the videos download would not be feasible. We finally found and decided to use *youtube-dl* [65], a small command line programme to download videos from YouTube that works in Python and is publicly released to allow users modifying it. It also allows to download the audio separately, which was also very convenient for extracting more easily the audio and their derived acoustic features, in case we decided to use them. We wrote a script which used youtube-dl to iteratively download all the videos in MP4 format by accesing the URL of each video, stored in a summary file for each type of query.

3.3.5 Preliminary Data Sets

During the process of downloading the videos and extracting and storing their metadata, we paid special attention to the distinction among the three types of queries, namely relevance, rating and view count. Our first idea was to keep only results from the relevance ordering because it provides a larger number of videos and the corresponding metadata are richer, what is very convenient for annotating the corpus through unsupervised learning methods. However, we found interesting to take into account the other two queries, since they can provide more videos to increase the size of the data set and, therefore increase the significance of the results.

Table 3.2 illustrates the total number of videos returned by each type of query at a preliminary state, i.e. before the filtering process. It can be observed that the number of videos retrieved when sorting by relevance is considerably bigger.

relevance	2,732 videos
rating	1,622 videos
viewCount	1,427 videos

Table 3.2: Number of videos returned by each query type.

However, rather than considering the results provided by each type of query as isolated data sets, we carried out a different strategy. We considered two different data sets or lists: List 1, which contains only the results provided by relevance and List 2, which gathers all the results. We can see the initial sizes of these data sets in Table 3.3.

	Types of query	**Initial size**
List 1	*relevance*	2,732 videos
List 2	*relevance + rating + viewCount*	5,781 videos

Table 3.3: Initial sizes of List 1 and List 2

3.4 Data Filtering

We have discussed in Section 3.2 the set of requirements we established for our corpus—or corpora—as we departed from two different lists, as explained in the previous section. Some of these require-

ments were fully satisfied at the data retrieval step, as it is the case of the date requirement, but the rest of them were only partially satisfied or there was not guarantee of their fulfilment. Therefore, in order to get this guarantee, we carried out a filtering process on the data. We divided this process into two parts, automatic filtering and manual filtering. The automatic filtering can only apply objective criteria and therefore it cannot fully satisfy a requirement such as ensuring that a video is a car advertisement. This is the reason why a manual filtering is required at the end of the process in order to guarantee the satisfaction of all the requirements.

3.4.1 Automatic Filtering

The automatic filtering performs the main part of the filtering process and reduces the data set almost to its final state. It was carried out through MATLAB scripts that read the input data from the metadata files produced in the retrieval process and filter them according to the conditions. These conditions are based on the requirements presented before and their objectives can be summarised as follows:

1. Delete repeated videos
2. Ensure a minimum amount of metadata
3. Minimise the effort to be done in the manual process to guarantee that every video is of the desired type.

In particular, in order to achieve objective 1, the videoIDs are used. Even though videos are retrieved through different queries, it may happen that the same video is returned by different queries. For example, a *SEAT*advertisement may be retrieved both by the *SEAT* query and by the *Volkswagen* query because both brands belong to the same company group or because of other multiple reasons. Regarding List 2, where videos were obtained according to a different way of sorting the results, repetitions of videos are even more frequent. Therefore, in order to avoid this inconvenience, we left only one occurrence of the videos with the same ID as part of our data set.

Regarding the second objective, we followed the constraint presented in the requirements: every video must have at least three raters. The idea behind this constraint is based on the way we derive the polarity labels of ground truth for later analsing the correlation with visual features. This procedure, which will be explained in detail later, consists in deriving the labels from metadata such as the number of likes, number of dislikes or the number of views. These metadata are provided by users as a feedback that describes how positively or negatively they perceive the video. Therefore, in order to be able to derive these labels, a minimum feedback must be required. Before taking the final decision, we assessed different possible solutions. For example, one option was to require either three raters or two comments, at least. The comments can be useful to extract a polarity label by using natural language processing techniques. However, we finally decided to set the stronger constraint of requiring always at least three raters because it guarantees certain uniformity in the corpus, while accepting videos without raters, but with some comments, might not be a guarantee of having enough feedback in terms of the perception polarity. The decision of setting three raters as the minimum lies on the idea of maximising the number of instances in the corpus, but with enough feedback to derive a label. If only one rater was allowed, there would not be enough feedback, and if only two raters were allowed, it could lead to a tie and hence ambiguity with little information. Therefore, we set three as the minimum.

For achieving objective 3, a couple of procedures were carried out. The idea of this objective is trying to remove from the list any video which is not a recent, professional car advertisement, which is the domain we defined in sections 3.1 and 3.2. First, we performed a quite objective operation that applies to one part of the requirement, deleting videos which are not advertisements. It is straightforward to realise that a TV advertisement has a limited duration. Therefore, although we cannot guarantee that every video that passes the filter is an advertisement, we do delete many videos that, with great certainty, are not advertisements. In order not to eliminate desired videos, we preferred to set a loose constraint: videos with a duration within 10 and 115 seconds are kept,

otherwise, they are removed from the lists. It is important to note that many advertisements uploaded to internet sites, such as YouTube, are longer versions of the spots which are played on television. The second measure we take in relation to this objective has the aim of complementing the previous one. It tries to retain advertisements, rather than discarding those which are not. To do so, the filter looks at the video titles and description and preserve only those videos with any of the key words. The key words used in this filter are very similar to the ones used to query the videos: "spot", "campaign", "publi*" and "advertisement".

To put it briefly, the whole automatic filtering process consists of four filters or steps: a deletion of repeated instances, a duration filter, a metadata filter and a key words filter. A summary of the operation of the filtering algorithm implemented on MATLAB is shown in Algorithm 2 to better illustrate the process.

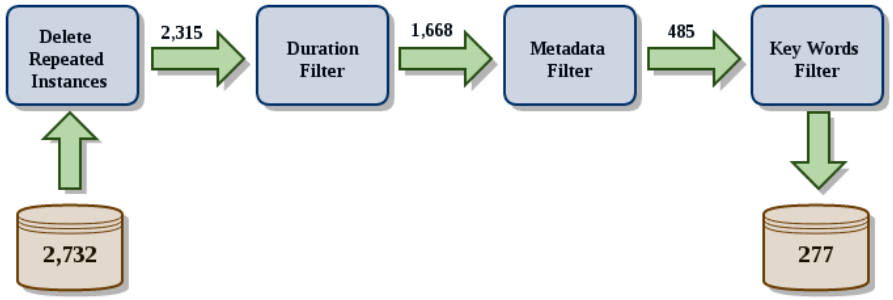


Figure 3.1: Size of **List 1** after each step of the automatic filtering

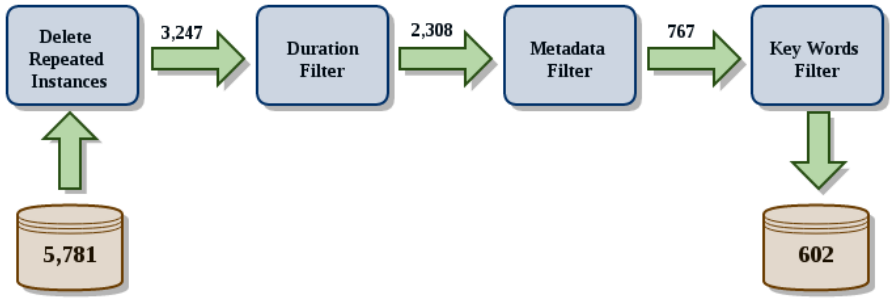


Figure 3.2: Size of **List 2** after each step of the automatic filtering

To illustrate how the automatic filtering modifies the lists, figures 3.1 and 3.2 show the size of the lists after each filter step of the process. The first figure corresponds to List 1, the one with videos retrieved by the relevance query, and the second figure corresponds to List 2, the one with all the results. Looking at these figures, one can extract several conclusions or observations. First of all, in the initial lists there are many repeated results, specially in List 2, something expected because there are a lot of overlapped results returned by the three types of queries. Nevertheless, by adding the results from rating and viewsCount, we achieved almost 1,000 more videos than with relevance after deleting the repeated instances. The duration filter considerably reduced also the number of instances from both lists, which was fine because we did not expect to get any advertisement with a duration out of the defined range, and the videos removed at this filter do not need to be manually checked. A very important observation is that the filter that reduces the lists most is the metadata filter, which removes 1,183 videos from List 1 and 1,541 videos from List 2. This can be regarded as a serious inconvenience as it is a great reduction. However, this filter is essential for the proposed approach to be successful. It is a fact that many videos do not have enough metadata to allow to extract polarity labels through unsupervised learning. Finally, the key words filter did not produced a significant reduction. First, because it is placed at the end of the process where the

lists have been already truncated. Second, because the key words are the same than for querying and normally the key words are present in the title or the description.

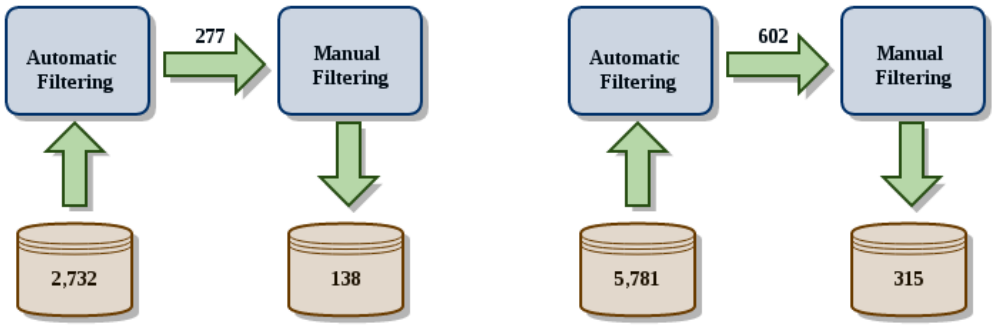
3.4.2 Manual Filtering

Although the automatic filtering does most of the work, it cannot provide a totally *clean* data set. It is essential for the research not to have a noisy data set, which is to say that we need to be sure that every video belongs to the defined domain. The importance of this lies on the fact that as a novel approach for extracting subjective information from visual features through a computational model, we would like our ground truth labels, derived from related metadata, to be as reliable as possible. Furthermore, since we annotate the corpus through unsupervised learning techniques, the importance of having a clean list is even greater. However, achieving this with automatic tools becomes quite complex and unreliable, reason why and a manual filtering becomes recommendable.

The manual filtering process consisted of two steps. First, we made an inspection of the video titles of all the instances that passed the automatic filtering. Looking at the title, it is easy to discover many videos that are not car advertisements. This procedure reduces significantly the effort to make at the second step. This second step consisted in watching all the videos and deleting those that were not car advertisements like the ones that can be watched in television. Despite it was stated that, as a requirement, videos should be Spanish, after watching them, some videos from South American countries and some from other European countries were also allowed under the criterion that they were really similar to the car advertisements published in Spain.

There were many different types of videos that passed the automatic filtering, but were deleted in the manual review. The most common ones were old car advertisements that had been published after 2010. There were also advertisements with bad quality, amateur advertisements, advertisements of other type of products and other types of videos related to cars. Repeated videos, although not detected by the automatic filter because they had different videoIDs, were also deleted.

Figures 3.3a and 3.3b show the sizes of the lists after the whole automatic filtering process and after the manual deletion of instances. We observe that approximately half of the instances that passed the automatic filtering had to be manually deleted.



(a) Size of **List 1** (b) Size of **List 2**

Figure 3.3: Sizes of lists 1 and 2 after the automatic and the manual filtering

After the manual filtering, we have the final configuration of the data sets in terms of size. The size of the data sets is a decisive parameter when evaluating the confidence of the results. The obvious observation is that filtering produces a large-scale reduction of the lists, only 5% of the instances retrieved at the first stage take part of the final lists. However, in any case, it is a necessary and inevitable procedure. We can be now sure that the videos that form the data sets satisfy the minimum requirements to be part of the experimentation process. Another important observation is that by merging the results from the three types of queries, a list with more than

twice the size of List 1 is achieved, which is a considerable improvement in terms of the confidence ranges that can be obtained when evaluating the classification results.

List 1	138 videos
List 2	315 videos

Table 3.4: Definitive sizes of both lists

3.5 Automatic Annotation of the Corpus

One of the most novel and challenging characteristics of this research project is the annotation of the corpus through automatic procedures. To the best of our knowledge, previous works on automatic assessment of aesthetic quality and video sentiment analysis made use of either already annotated corpora or carried out specific user surveys to get the labels. Let us examine the procedures in this respect of the most relevant related works. For instance, Datta *et al.* [9] made use of the online photo sharing community *Photo.net* as data source. It provides two types of scores from users: originality and aesthetics. They also retrieved from each picture the number of times viewed by users and the number of peer ratings. However, they finally used only the average aesthetics score as ground truth for classification purposes. Similarly, in a later related work [29], the same data set and annotations were used. In [2] they used a large data set of videos which had been specifically ranked by 10 individuals for a challenge on the topic. Another important work on computationally assessing the visual quality of videos was carried out by Moorthy *et al.* in [32], where they prepared their own controlled user study by recruiting 33 participants who rated the aesthetic appeal of 40 videos. In [57] the same data set and labels were used for trying to improve the results. In the field of video sentiment analysis, Morency *et al.* [33] and Pérez Rosas *et al.* [45] annotated the polarity of the opinions expressed on the videos themselves.

In this work we propose to derive the polarity annotations of the videos through unsupervised learning techniques instead. The idea, which will be explained in detail later, basically consists in inferring a label for each video related to how positive or negative it has been perceived by users by using the metadata extracted from the videos and exposed in Section 3.3. This procedure presents some disadvantages. First of all, it is a much more complicated method because it involves additional operations and deeper data analysis. Second, methods of unsupervised learning usually lead to less accurate and reliable results. In spite of these drawbacks, there are a number of advantages that make unsupervised learning be worth using it:

- It is an inexpensive procedure, since it is not necessary to recruit a group of people that provide their scores for the videos, which, apart from being an expensive method, the viewers provide the scores under a controlled environment which may affect their opinions.
- Rather than getting scores from a group of viewers specifically selected for the experiment, we obtain labels from the metadata that actual viewers and users of the videos provide. This is much more interesting in terms of the application of the results of this project since we can be sure that the annotations are based on the way potential consumers perceive the advertisements.
- The procedure we propose involves deriving the labels from several metadata, rather than from a single parameter. This offers a wider definition of the subjective information we try to infer through computational methods. Instead of assessing a score based on aesthetics as beauty, we could potentially define a more general assessment model of how users perceive the videos.
- Since we use computational methods of unsupervised learning, we will be able to define different strategies that might slightly modulate this sense of perception.

3.5.1 Introduction to Unsupervised Learning

Machine learning is a field of artificial intelligence which seeks to extract information from some input data and apply and generalise the learnt knowledge to unseen data instances. The learning process is performed through machine learning algorithms which, at a general level, can be divided into supervised and unsupervised learning methods. The difference between both is the domain on which they can be applied, which, in turn, depends on the available input data. Supervised learning is used when the available data consists of some input $\mathbf{x}$ and their corresponding labels or outputs $\mathbf{y}$. On the contrary, unsupervised learning is applied when only the input $\mathbf{x}$ is known and there is no known information about the labels. For instance, if the machine learning problem is coin value recognition by means of characteristics such as size and mass from a set of sample coins, supervised learning methods would be used if besides the coin characteristics, the actual values of the sample coins (training instances) were known. Otherwise, it would be a case of unsupervised learning.

Supervised and unsupervised learning algorithms are quite different and, due to the lack of information, unsupervised learning problems are usually much more challenging. One of most common techniques of unsupervised learning is cluster analysis, which consists in methods for partitioning the data into clusters or classes according to intrinsic patterns that an algorithm tries to discover. One of the most celebrated clustering algorithms is k-means clustering, first introduced by Lloyd in [26].

k-means Algorithm

k-means clustering partitions input data into k mutually exclusive clusters. The most important singularity of this algorithm is that it operates on actual instances, instead of hierarchically computing dissimilarity measures as hierarchical clustering methods, so it creates a single level of clusters and it is more suitable for big data sets. The idea behind this algorithm is to cluster instances in such a way that objects within a cluster are as close to each other as possible, and as far from objects belonging to other clusters as possible. Each cluster is defined by a centroid, which is the point to which the sum of distances from all objects in that cluster is minimised. The notion of distance is a key concept for the operation of the algorithm, since different distance measures can be used to determine the clusters. A more detailed discussion of the distance measures that have been used in this work will be offered in Section 3.5.4. For a given set of d-dimensional input data $(x_1, x_2, \ldots, x_N)$, a k-means algorithm will try to partition the N observations into $k \leq N$ clusters $C = \{C_1, C_2, \ldots, C_k\}$ according to the following statement:

$$\arg\min_{\mathbf{C}} \sum_{i=1}^{k} \sum_{x_j \in C_i} d(x_j, \mu_i) \tag{3.1}$$

where μ_i is the centroid in cluster C_i and d represents the distance measure. Regarding the operation of the algorithm, it uses an iterative method that moves instances between clusters until the sum in Equation 3.1 cannot be decreased further. This algorithm has a MATLAB implementation [31] that allows setting different distance measures or changing parameters such as the number of random repetitions to find better local minima.

3.5.2 Metadata for Clustering

From all the metadata specified in Table 3.1 and retrieved from the videos through the YouTube API, some are useful for filtering purposes (duration, publish date, title or description), for organising the data set (videoID), for downloading the videos (URL) or for many other purposes. One of the main functions of these metadata is to be used for performing the cluster analysis, as explained in the previous sections. However, not all the metadata are useful for deriving polarity labels.

Along this section, we will describe those metadata which are potentially useful for the clustering and define two derived, new metadata.

The characteristic of a metadatum to be useful for the cluster analysis is that it must reflect in some way the feedback provided by users in terms on how the perceive the video. The metadata that can potentially describe the appeal of a video to users are the following:

- **viewsCount**: the **number of views** is the total amount of times a video has been played either in youtube.com or through an embedded player in a different web page. YouTube includes mechanisms to avoid frauds such as playing a video repeatedly and it only increments the counter when a few seconds have been played. This metadatum can be of great interest for deriving the labels because it implicitly carries information of how good or bad a video has been received by users. We hold the reasonable assumption that the greater the number of views is, the better its assessment should be. However, this might be sometimes a noisy parameter because of the recurrent phenomenon of viral videos on the internet as well as it can happen that a potentially good video is not spread enough.

- **favoriteCount**: the favourite count is the number of times a video has been selected by a user as a favourite video. It should potentially reflect the assessment of a video, as the greater this quantity is, the better the assessment should be. However, in practice, at least in the domain of car advertisements, favoriteCount is most often zero. Therefore we have discarded this metadatum regarding the clustering.

- **numComments**: the **number of comments** a video has can also be used to derive labels related to the video assessment. Our hypothesis is that, in principle, in our videos domain, the more comments a video has, the better the assessment. Of course, there might be exceptions and a video could have many comments because of a negative characteristic. However, in any case, within the car advertisements domain this is not a generalised behaviour.

- **numRaters**: the **number of raters** is the amount of YouTube users who have rated a video or clicked in the *like* or the *dislike* buttons. Although it might be probably not the most informative metadatum, it is still quite interesting for assessing the perception of a video since, similarly to the number of comments, it can be assumed that a high number of raters generally contributes positively to the assessment of the video.

- **rating**: the rating is actually the **average rating** that have been provided by users. This is a special metadatum because it was introduced to reflect the old way YouTube users had to value a video. Until March 2010 [63], instead of a *like* and a *dislike* button, there was a system consisting of five stars from which users could choose from 1 to 5 in halves of star. However, YouTube changed this system because they considered that it did not reflect a real 1-to-5 rating, but just a binary assessment, as it is posted in the official YouTube forum [59, 62, 63]. Hence, the star-based system was replaced by a simpler likes/dislikes system. An example of the old interface can be seen in Figure 3.4.

- **numLikes**: the **number of likes** is simply the number of times the users have clicked the like button. This system has worked only since March 2010. It is very interesting for deriving polarity annotations because it clearly follows a "more is better" criterion.

- **numDislikes**: similarly, the **number of dislikes** is the number of times the users have clicked the dislike button. It behaves opposed to the number of likes, the more number of dislikes a video has, the worse the assessment should be. Therefore, this metadatum is of great interest for deriving polarity labels.

Looking at the values of these metadata on the YouTube videos, one can realise that a generalised behaviour of users is that they often prefer indicating what they like, rather than indicating what they do not. Therefore, it is rare to find videos with a very bad valuation, whereas it is quite common to find videos with extraordinaryly high assessment. Videos that users do not like have normally no likes or a few of them and several dislikes, but not many, together with a low number

Figure 3.4: Old YouTube interface with the star-based rating system

of views. Therefore, perhaps we should talk of good videos and *less good* videos, instead of good and bad.

In addition to the described metadata, we found interesting to create two new, derived metadata either to simplify or to *improve* some of the raw YouTube metadata in order to make the cluster analysis more effective.

Likes-Dislikes Ratio

The main reason for the creation of this new metadatum is to merge the $numLikes$ and the $numDislikes$ into one single metadatum. Mainly to prevent possible problems due to data sparsity, it is more convenient for the k-means algorithm to deal with fewer features. Therefore, we derive the likes-dislikes ratio ($ldRatio$), which represents the proportion of likes from the total number of votes, i.e. likes and dislikes, and is computed as follows:

$$\mathbf{ldRatio} = \begin{cases} \frac{numLikes}{numLikes+numDislikes} & numLikes + numDislikes \geq 0 \\ 0 & numLikes + numDislikes = 0 \end{cases} \tag{3.2}$$

Its interpretation in terms of the influence on the assessment is similar to the one of the number of likes: the higher, the better the assessment. By definition, its value must lie between 0 and 1.

View Count Score

It has been observed that the range of values of $viewsCount$ is very wide and this makes its dispersion terribly huge. This would not be such bad news if the wide range reflected the real differences among the assessment of the videos. However, in practice, the differences in the values of $viewsCount$ do not necessarily coincide with a logical assessment. For instance, one video might have 500,000 views for some reason, perhaps *virality*, but this fact does not probably mean that it is ten times better than a video with 50,000 views, which is also a very high number in terms of views. To minimise the effect of this behaviour, we derive $viewsCountScore$, which is a new

metadatum that maps the number of views into a score from 1 to 5, according to ranges based on the percentiles of the distribution of data:

$$\textbf{viewsCountScore} = \begin{cases} 1 & 0 \leq viewsCount < 750 \\ 2 & 750 \leq viewsCount < 2,000 \\ 3 & 2,000 \leq viewsCount < 5,000 \\ 4 & 5,000 \leq viewsCount < 15,000 \\ 5 & viewsCount \geq 15,000 \end{cases} \tag{3.3}$$

Its interpretation in terms of the influence on the assessment is similar to the one of the number of views: the higher, the better the assessment being 1 the minimum value and 5 the maximum.

3.5.3 Clustering Strategies

Once we defined the two new metadata and after examining at the available metadata, we realised that there were significant differences among them. For example, we have explained before that the number of views might be useful for deriving polarity labels because we hold that, apparently, if a video is very appealing, it is usually shared among users, it appears in many different web sites, etc. Therefore, the higher the number of views a video has, the better the assessment should be. However, the nature of this metadatum is rather different to, for instance, the nature of the number of likes. The effect of the latter on the video assessment is easier to interpret, as an increment of this parameter directly implies that someone has positively appreciated the video and thus has clicked the *like* button. The difference between these metadata lies on the way they are provided by users: some are explicitly provided (number of likes), while some are implicitly provided (number of views). Not only is this difference important because of their diverse nature, but also because we hypothesise that these types of metadata represent different user profiles as well. On the one hand, there are users who explicitly express their opinion and, on the other hand, users who normally do not explicitly express their opinion, but whose assessment is implicitly provided as they watch videos. Based on these hypotheses, we define the following types of metadata:

- **Explicit-opinion metadata**: these metadata require the active participation of users. That is to say, these metadata represent those users who, besides automatically contributing to the video assessment by watching it, they do explicitly express their opinion. Apart from giving their opinion by clicking the *like* or the *dislike* button (or by clicking on the star-based rating in the past), users can also explicitly express their opinion by writing a comment. Therefore, from the metadata we have presented in the previous section, the ones that can be classified into this type are the following: **favoriteCount**, **numComments**, **numRaters**, **rating**, **numLikes**, **numDislikes** and **ldRatio**.
- **Implicit-opinion metadata**: these metadata are provided automatically by users. That is, as soon as a YouTube user watches a video, these metadata are incremented, automatically contributing to the implicit assessment of the video. This type of metadata represent those users who watch videos without explicitly providing their opinion about them. However, even if they do not leave any rate or comment, they implicitly contribute to the overall video assessment. The only metadata of this type from the ones defined in the previous section are **viewsCount** and **viewsCountScore**.

We consider this classification of metadata a very important differentiation for the cluster analysis. Since clustering is a delicate operation, we thought that mixing all the metadata could not lead to good clusters and therefore to good convenient results. We believe that a good practise is to carry out separate cluster analysis with each type of metadata and with the combination of both in order to check this hypothesis and the performance of each group separately. For the same purpose of distorting the clustering the less possible, we decided to remove all metadata that do not contribute in a noteworthy way to the analysis: numLikes and numDislikes are removed because they are better expressed by ldRatio; viewsCount is removed and substituted by viewsCountScore and

favoriteCount is also discarded because it is not useful in the car advertisements domain. Hence, with all this information, we define three different clustering strategies as follows:

S_1 **explicit-opinion metadata**	**S_2** **implicit-opinion metadata**	**S_3** **combination of both**
ldRatio rating numComments numRaters	viewsCountScore	viewsCountScore ldRatio rating numComments numRaters

Table 3.5: Clustering strategies in terms of metadata

These three clustering strategies are applied both on List 1 and List 2 and each of them might potentially create a particular annotation of the corpus. Although this approach adds extra complexity to the experiments, this distinction will allow us to carry out separate analyses of the results from the different strategies and evaluate which strategy performs better. An easiest alternative would have been to perform the cluster analysis on the complete set of metadata, but in such a case we would have kept the doubt of another combination performing better.

3.5.4 Cluster Analysis Setup

As it has been mentioned in Section 3.5.1, the unsupervised learning algorithm that was chosen for obtaining the annotations for the corpus is the well-known k-means clustering algorithm. In an early state of the research process, we started using WEKA[3] machine learning software [14] to perform the cluster analysis because it provides a nice interface that allows doing simple tests and obtaining some preliminary labels for the videos, which were very useful to get an idea of how our expectations for the research could be. However, having defined different strategies and with an interest in trying multiple combinations, the implementation of the k-means algorithm that MATLAB offers [31] became a more practical alternative, mostly because it allowed us to automate all the process and easily store all the results.

Besides the three different combinations of metadata offered by the clustering strategies defined in the previous section, we did not want to limit our analysis to a single configuration of the k-means algorithm. Therefore, we decided to create several clusters by adjusting different parameters of the clustering algorithm and testing all the possible combinations. In spite of increasing the complexity of the analysis, this procedure explores more possibilities, which, on the one hand, allowed us to find an optimal solution and, on the other hand, enabled us the evaluation of the behaviour of the different approaches to provide a richer analysis of the results. It is important to remark that the different combinations we explored might very possibly generate dissimilar annotations. The parameters of the k-means algorithm that were modulated are the number of classes and the distance measure for the clustering. Furthermore, we have also evaluated the resultant clusters when computing the z-scores of the metadata.

Number of classes

As it has been explained in Section 3.5.1, the k-means algorithm partitions the data set into k independent clusters or classes. This means that one can force the algorithm to create a determined number of groups of data. In the literature on aesthetic quality assessment of videos and images [47, 9, 32, 57, 2] as well as in sentiment analysis works [33, 56] the usual procedure is to simplify the problem by reducing it to a binary or two-classes classification task, whose corresponding interpretation is much more straight forward and clearer. Nevertheless, we found three reasons to

[3] An introduction to this software can be found in Appendix A

test on more than 2 classes. First of all, it is quite interesting from the research point of view to evaluate classification results when generating several classes of labels. In addition, it supposes an important differentiation to previous works and, finally, since we automatically generate the annotations through clustering techniques, it is good practise to evaluate the k-means algorithm with more than 2 classes because it might happen that the quality of the resultant clusters, i.e. how well-separated they are, is better with other number of classes than 2.

Therefore, we have implemented our algorithm to generate labels by running the k-means algorithm from $k = 2$ up to $k = 5$. In principle, when the data set is divided into only 2 classes, it is reasonable to think one class could correspond to *good* or positive videos and the other class to *bad*, *less good* or negative videos. If there are 3 classes, we could suppose that classes correspond to positive, neutral and negative. Following the reasoning, the for 4 and five classes, these could represent more particular levels of users satisfaction, for instance *bad*, *below average*, *average*, *good* and *very good*. As it can be observed, distinguishing among up to 5 classes provides granularity to the model, which, from the applications point of view, is a very interesting improvement, compared to previous works. However, it is important to remind that it is possible that the resulting classes may not have such a logical interpretation.

Distance measure

The basis of the k-means algorithm for partitioning a p-dimensional data set into k clusters is to group data according to the distance between them, so that instances within a cluster are as close to each other as possible. Consequently, in order to determine how close to each other the examples of the data set are, different similarity measures can be used and the resultant clusters will have a strong dependence on this measure. More exactly, each cluster is defined by its member instances and by its centroid, or centre. The centroid for each cluster is the point to which the sum of distances from all instances in that cluster is minimised and centroids are computed differently for each distance measure. From the distances measures available to be used with the MATLAB k-means algorithm, we have selected the following:

- **squared Euclidean distance**: it is the square of the *ordinary* Euclidean distance. Although it is not a distance metric because it does not satisfy the triangle inequality [21], it is perfectly usable for comparison purposes, as it is the case. When the squared Euclidean distance is used, each centroid becomes the mean of the points in the cluster.

$$sqEuclidean(x, x') = \sum_{k=1}^{d} (x_k - x'_k)^2$$

- **city block distance**: also known as Manhattan distance, it is the sum of absolute differences (SAD). Each centroid becomes the component-wise median of the points in that cluster.

$$cityblock(x, x') = \sum_{k=1}^{d} |x_k - x'_k|$$

- **cosine**: it is not a distance metric either. For computing this measure, data points are treated as vectors and, in MATLAB, it is defined as one minus the cosine of the angle formed by the two points. Each centroid is the mean of the points in that cluster, after normalising those points to unit Euclidean length.

$$cosine(x, x') = 1 - cos(\theta) = 1 - \frac{x^t x'}{\|x\|\|x'\|}$$

- **correlation**: this similarity measure is defined as one minus the sample correlation coefficient between points, when treated as sequence of values. In this case, the centroids become the component-wise mean of the instances in the cluster, after centring and normalising them to zero mean and unit standard deviation.

$$correlation(x, x') = 1 - r_{xx'} = 1 - \frac{\sum_{k=1}^{d}(x_k - \bar{x})(x'_k - \bar{x}')}{\sqrt{\sum_{k=1}^{d}(x_k - \bar{x})^2 \sum_{k=1}^{d}(x'_k - \bar{x}')^2}}$$

Z-score normalisation

We have pointed out before that clustering procedures are delicate methods because they involve many parameters and the unsupervised nature adds much uncertainty. Therefore, every effort which can be done to try to improve the performance is not in vain. For this reason, we decided to evaluate the same strategies defined in 3.5.3, but after a normalisation of the metadata. As stated by [18], it is a good practise, before performing a cluster analysis, to compute the Z-scores of the data involved in the process, specially if the ranges of values differ among them. The Z-score of a datum, also known as standard score, consists in shifting the values towards the position of a standard normal distribution ($\mu = 0$ and $\sigma = 1$), so it is defined as follows:

$$z = \frac{x - \mu}{\sigma}$$

However, distances *cosine* and *correlation* perform, by definition, an intrinsic normalisation of the data for determining the clusters. Therefore, it is meaningless to compute the Z-scores when the cluster analysis is configured to use these distance measures. Consequently, we have computed the Z-scores of all the metadata for the cases when the distances are *sqEuclidean* and *cityblock* before performing the cluster analysis.

As a summary of all the clustering configurations, we can observe that the combinations of all of them lead to multiple possible annotations or data sets. More exactly, metadata is combined following 3 different strategies, which are the input of the cluster analyses. For each strategy, the k-means algorithm is run for 4 values of k, from $k = 2$ to $k = 5$. Finally, each of these configurations is evaluated for the 4 different distance measures, either after a Z-score normalisation or not. Therefore, there is a total of $3 \times 4 \times 4 = 48$ clustering combinations, each with potentially different annotations. Furthermore, it is important to remind at this point that we are carrying out these operations on two lists, List 1 and List 2. So at the end of the clustering process, there are 96 combinations. From now on, we will call these combinations with potentially different annotations *data sets*, although there are indeed only two different corpora, List 1 and List 2.

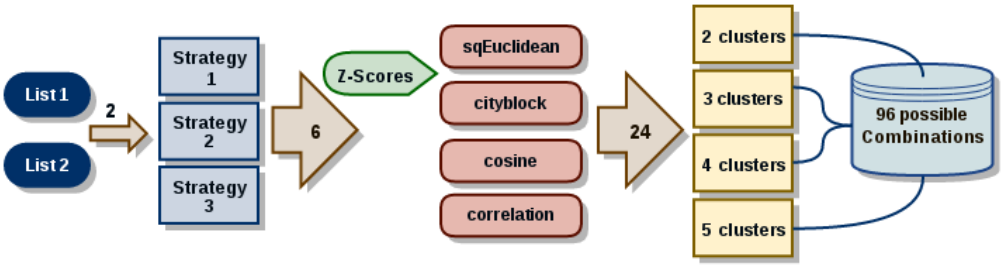


Figure 3.5: Steps of the cluster analysis process with the number of combinations after each stage

However, due to the nature of the k-means algorithm, it might happen that some configurations lead to empty clusters during the process, which is not a valid solution. This is particularly usual

when the number of classes is higher and when there is only one metadatum, as it is the case of S_2, which uses only viewsCountScore. Furthermore, the correlation measure is not valid for any combination of S_2. Thus there are many non-valid combinations, so the real number of data sets at the end of the clustering process is less than 96. The number of valid combinations for each list and strategy can be seen in Table 3.6

	S_1	**S_2**	**S_3**	**Total**
List 1	16	10	16	42
List 2	16	10	16	42
Total	32	20	32	84

Table 3.6: Number of valid clustering combinations that lead to potentially different data sets for each list and strategy

Chapter 4

Visual Features Extraction

Visual features are perhaps the nuclear part of this research project. We may best describe the aim of this work as evaluating how well visual features can predict the assessment of a video. Therefore, visual features play a crucial role in the analysis and everything else is around them to help carry out all the required operations.

We begin this chapter with a brief introduction to the type of features that have been extracted and a motivation for their selection and usefulness. Then, the main part of the chapter is dedicated to explain in detail each feature, their applications and the procedures for obtaining them. Finally, in Section 4.7 we include some comments related to the technical aspects for computing the feature values on the videos from the corpus.

4.1 Introduction

It is important to note that the visual descriptors we extract and on which we test the experiments are low-level video features. By low-level we mean features which are directly extracted from the digital representation of images and videos and do not have a very close relation to the way how people describe what they see, i.e. to semantics. In contrast, high-level features are much closer to semantics and concepts.

The decision of which visual features to extract and, in turn, to test how they correlate with the user perception of the video has been motivated and inspired by previous works, such as that from [9] and others, who proved the convenience of some descriptors for assessing the aesthetic value; we have also relied on our own experience, knowledge and research in photography and filmmaking, our own intuition and in some cases, some features have been extracted without any strong support, just for finding out if they have some hidden influence on the visual assessment, as it has been eventually happened.

Since one of the purposes of this work is to demonstrate the usefulness of low level visual features for assessing the user perception of videos, we have extended the meaning of some image-level features towards the temporal dimension, which is what distinguishes a video from a picture, by taking the average of the frame-level features along all the frames of the video (Equation 4.1) and computing the standard deviation of the distribution of the feature values in the frames (Equation 4.2). On the other hand, some other features are exclusive of videos and do not need a specific extension towards the temporal dimension. Altogether, we have extracted a total of 21 features.

Along this chapter, we present these features organised according to the visual aspect they describe.

$$avg(f_{video}) = \frac{1}{N}\sum_{n=1}^{N} f_n \tag{4.1}$$

$$std(f_{video}) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(f_n - avg(f_{video})^2} \tag{4.2}$$

4.2 Temporal Segmentation

The distinguishing characteristic of videos is the temporal dimension, thus, features describing this aspect are of great interest. In filmmaking or, by analogy, in advertisements-making, temporal segmentation has a crucial importance, since it is the basis of montage, the editing technique that allows the creation of most effects cinema produces [4, 37]. Montage not only creates many semantic effects, but quantitatively, the level of segmentation, i.e. the number of cuts, is a good indicator of meaning. For example, an action scene has usually a higher number of cuts than a calm, descriptive scene. In [4], David Bordwell says "when the shot durations are modified in relation to each other, the film-maker is playing with the potential of the rhythm on the montage. [...] In general, when controlling the rhythm in the montage, the film-maker adjusts the amount of time the spectators have to understand and think of what we watch. A series of fast shots, for instance, leave as few time to think about the scene."[1].

A temporal segmentation of a video implies to determine the transitions between subsequent shots. There exist different types of transitions in video composition, such as fades, dissolves, wipes or cuts. However, since most transitions in video advertising are abrupt cuts [4, p. 247], the study of features related to temporal segmentation have been based on this type of shot transitions. There exist, in the literature [23], different techniques for detecting abrupt transitions within a video, most of them based on the study of the difference between consecutive frames of a measure based on a descriptor. For our purpose we choose the sum of absolute differences (SAD) [58] of the grey intensity, which is defined for each frame n as follows:

$$D(n) = \frac{1}{H \cdot W}\sum_{x=1}^{W}\sum_{y=1}^{H}|I_n(x,y) - I_{n-1}(x,y)| \tag{4.3}$$

The performance of this measure can be improved by using a new measure $M(n)$ based on its second derivative, which offers additional robustness at high speed movements because its detects abrupt transitions of the first derivative of the descriptor:

$$M(n) = -D''(n+1) = -(D'(n+1) - D'(n)) \tag{4.4}$$

with

$$D'(n) = D(n) - D(n-1)$$

[1]Free English translation of the Spanish version of the book by the author of this report, due to impossibility for consulting the original text. Official Spanish quote: "Cuando el cineasta modifica la duración de los planos en relación unos con otros, está controlando el potencial *rítmico* del montaje. [...] En general, al controlar el ritmo del montaje, el cineasta regula la cantidad de tiempo que tenemos para comprender y reflexionar sobre lo que vemos. Una serie de planos rápidos, por ejemplo, nos deja poco tiempo para pensar acerca de lo que estamos viendo."

This measure is computed for every frame of the video and a threshold is required to decide if there is a cut at a certain frame or not. The threshold chosen after performing several tests with previously labelled videos is 0.18. Although a hundred percent accuracy cannot be always achieved, this algorithm achieves a good performance. An example of the operation of the cut detection procedure on a video from the corpus can be observed in Figure 4.1.

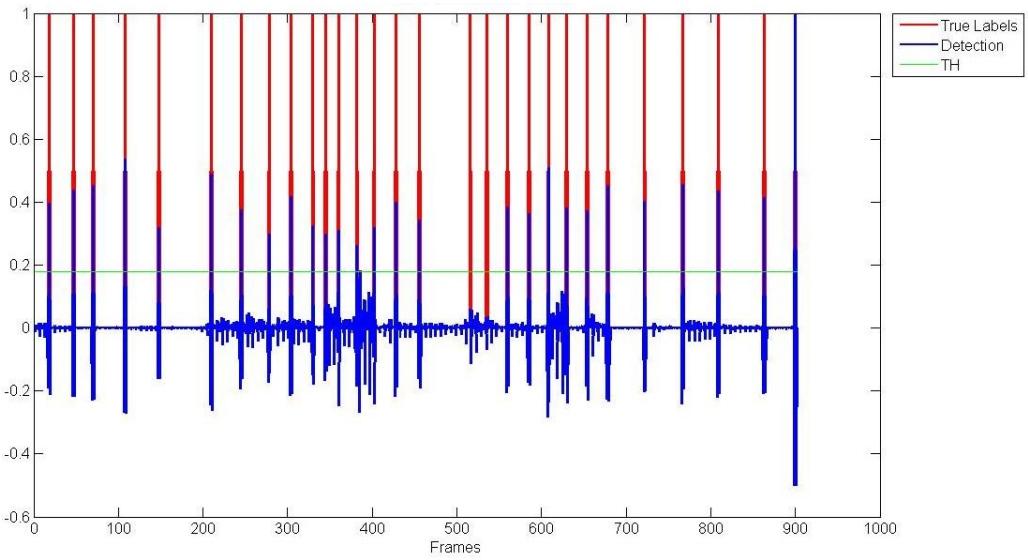


Figure 4.1: Cut detection performance on video -OfmscQNZ5c with SAD and measure 2der. TH=0.18

Having the temporal segmentation of the videos in terms of their abrupt shot transitions, i.e. cuts, we can define a set of features related with this aspect:

Absolute number of cuts

It is referred to as the total number of abrupt cuts within a video, without taking into account its duration or any other characteristic of it. It is indicative of the degree of segmentation of the spot, although it is not normalised by the duration.

feature ID	f_1
short name	num-cuts
theoretical min	0
theoretical max	∞
min	2
max	118
mean	25.79
standard deviation	17.87

Table 4.1: Statistics and information of '**absolute number of cuts**'

Longest shot

It is the duration, in seconds, of the longest shot, considering shots to be the fragments of video between two cuts, between the beginning of the video and first cut and between the last cut and the end of the video. It does not take into account the total duration of the video either. Shots with

a very long duration are usually indicative of an intention to slow down the spot rhythm or have some other particular meaning, while a short longest shot is indicative of fast rhythm throughout the whole spot.

feature ID	f_2
short name	longest-shot
theoretical min	0
theoretical max	∞
min	1.32
max	63.73
mean	6.51
standard deviation	6.52

Table 4.2: Statistics and information of '**longest shot**'

Average shot duration

It represents the average duration of the shots of the video, in seconds. The interpretation of this feature is inverse to that of f_1 (num-cuts) and in this case the average implies a normalisation. A low value means high degree of segmentation which is usually indicative of a fast rhythm and vice versa.

feature ID	f_3
short name	avg-shot-duration
theoretical min	0
theoretical max	∞
min	0.33
max	20.36
mean	1.93
standard deviation	1.90

Table 4.3: Statistics and information of '**average shot duration**'

Standard deviation of shots duration

The standard deviation of the duration of the shots within a video might be useful to measure how different in duration are the shots along the video. An example of this is that it is not infrequent that an advertisement is segmented according to its music beats, what yields uniformity in the shots and therefore, this effect can be captured by a very low standard deviation. On the contrary, there are spots with different parts, some with long shots and some with short shots, what yields a higher standard deviation.

feature ID	f_4
short name	std-shot-duration
theoretical min	0
theoretical max	∞
min	0.33
max	34.33
mean	1.88
standard deviation	3.20

Table 4.4: Statistics and information of '**standard deviation of shots duration**'

Cuts per minute

This feature is a normalisation of the absolute number of cuts, with respect to the duration of the video, i.e. the absolute number of cuts divided by the duration of the video and multiplied by 60. It represents the density of the cuts, whatever its duration is. This feature can be seen as a version of f_3 (avg-shot-duration), since they behave inversely, but represent the same effect. If the average shot duration is big, the average cuts per minute will be low and therefore their interpretations will be also opposed.

feature ID	f_5
short name	cuts-per-min
theoretical min	0
theoretical max	∞
min	1.94
max	145.46
mean	37.78
standard deviation	19.59

Table 4.5: Statistics and information of '**cuts per minute**'

4.3 Intensity

Intensity in a still image is referred to as the average value of the pixels of the grey-scale version of the image. In photography and film-making, intensity is also referred to as brightness and it is usually controlled to capture *correctly* exposed images, in terms of the useful exposure range of the film or sensor. However, on the one hand, the exposure is not the same under daylight conditions than inside and, on the other hand, the exposure does not have to be necessarily the *correct* one, but many effects can be created by under- and overexposing the image. For instance, David Bordwell points out in his book *Film Art* [4, pp. 186–189] that American black cinema in the 40's used to underexpose certain dark parts of the image to harmonically match other illumination techniques over the rest of the shot, while other films used overexposure to create particular effects. He gives the example of *Ordet* (Carl Dreyer, 1965), where overexposure is used to create mystical environments. Furthermore, this feature not only captures effects of exposure, but also the predominance of dark or bright colours and objects in the video.

In order to obtain a feature related to the video intensity or brightness, we take the mean of the pixel values of the greyscale version of all the frames of the video. Then, we can extend this image-level feature to the video level by computing the average intensity and the standard deviation along all the frames of the video. When computing this values, we do not take into consideration the black frames, i.e. 0-intensity frames, that usually come before and after the content, so that they do not distort the features values of the content. Note that the duration of the videos is quite short and the mean value of the features can be highly affected by three or four seconds of blackness.

A summary of the procedure and basic operations for extracting these features related to intensity can be found in Algorithm 3 in Appendix B.

Average intensity

By taking the average of the intensity along all the frames of a video we obtain a feature which indicates how its brightness is in general, i.e. we may be able to say if it is a dark or a bright video.

feature ID	f_6
short name	avg-intensity
theoretical min	0
theoretical max	255 (8-bit coding)
min	16.51
max	236.30
mean	87.11
standard deviation	35.71

Table 4.6: Statistics and information of '**average intensity**'

Standard deviation of the intensity

The standard deviation of the distribution of the frame intensities along a video is indicative of the uniformity of the brightness within the video. A low value of the std-intensity can be indicative of a video with few changes of intensity in the frames, which may mean semantically the use of the same scenes along the whole video, for instance.

feature ID	f_7
short name	std-intensity
theoretical min	0
theoretical max	$180.31 = \frac{1}{\sqrt{2}} \times 255$ (8-bit coding)
min	1.65
max	97.63
mean	37.42
standard deviation	14.73

Table 4.7: Statistics and information of '**standard deviation of the intensity**'

4.4 Entropy

In information theory, the entropy is a statistical measure of the randomness of a variable. Applied to image processing, the entropy can be used to characterise the randomness of the pixel values and thus, model in some way the texture of the image. The entropy of an 8-bit coded image is defined as follows:

$$E = -\sum_{b=1}^{256} p \cdot \log_2(p) \tag{4.5}$$

where, applied to images, p is the histogram count for each bin (with 8-bit coding there will be 256 bins), treating the RGB image as a multidimensional greyscale image. The texture of an image can be interesting because it gives an idea of its complexity, which can help transmit one effect or another to the spectator. From this measure, we derive the following set of features:

Average entropy

It is the average entropy along all the frames of the video, without taking into consideration the black frames to prevent from distortions in the value. This simple feature can offer a general idea of the complexity of forms and textures in the video.

feature ID	f_8
short name	avg-entropy
theoretical min	0
theoretical max	$8 = \log_2(256)$ (8-bit coding)
min	2.74
max	7.56
mean	6.18
standard deviation	0.82

Table 4.8: Statistics and information of '**average entropy**'

Standard deviation of the entropy

The standard deviation of the entropy along all the frames of the video, without taking into consideration the black frames, represents the variation of the texture within video, which may be indicative of monotony in the forms if the value is low and vice versa.

feature ID	f_9
short name	std-entropy
theoretical min	0
theoretical max	$5.66 = \frac{1}{\sqrt{2}} \times 8$ (8-bit coding)
min	0.12
max	3.39
mean	1.49
standard deviation	0.73

Table 4.9: Statistics and information of '**standard deviation of the entropy**'

Percentage of low entropy frames

We observed that most advertising videos insert some frames showing the brand logo, descriptions of the car or conditions of the offer among the filmed scenes or at the end. These frames have usually a monochromatic background (e.g. black or white) and letters or signs in the front. This kind of frames, with a large portion of the frame in a single colour, are particularly characterised for having very low entropy by comparison with filmed frames. After several tests with different videos a threshold of 2.85 entropy value was established to determine whether a frame was of this type. Therefore, this feature describes the percentage of frames, without taking into account the black frames, whose entropy is below the above mentioned threshold. This value will give an idea of the portion of advertising video which is composed by these special frames, which normally are not filmed scenes, but descriptive written information and signs.

feature ID	f_{10}
short name	pct-low-entropy-frames
theoretical min	0
theoretical max	1
min	0
max	0.54
mean	0.09
standard deviation	0.09

Table 4.10: Statistics and information of '**percentage of low-entropy frames**'

Low entropy end

This is a binary feature that determines if the end of the video is mainly formed by low entropy frames, that is, frames with a monochromatic background, as described in the previous feature. Many advertising videos end with this kind of frames, while some of them do not and others insert a filmed shot before finishing the video, making the difference. We think that it could be interesting to design a feature to automatically distinguish which videos make this practice and which do not. For computing this feature, the end of the video is considered to be the last 10% of the total frames and in order to decide if the end is formed by low entropy frames, the 85% of the frames belonging to the end must have a value of entropy below the threshold (2.85).

feature ID	f_{11}
short name	end-low-entropy
possible values	0 / 1
instances with 0	79%
instances with 1	21%

Table 4.11: Statistics and information of '**low entropy end**'

4.5 Colour

It is not news that colour plays a very important role in photography and filmmaking and there are theories about psychology and colour, apart from the commonly established relations between some colours and different concepts or ideas. Particularly in cinematography, after the establishment of colour in films in the 50s, directors and photographers have taken advantage of the properties of colour for creating effects. David Bordwell points out the importance of colour on the *mise en scène* in [4, pp. 148–157,186–189] as one of the most effective resources in filmmaking. As a way of illustration with a recent example, we can look at the TV series *Breaking Bad* (2008–2013, Vince Gilligan), in which colours play a crucial role in transmitting emotions and developing characters, as it is shown in the picture posted in [50].

4.5.1 Hue and Saturation

A well-known and widely used colour model is HSV [48]. It represents colour using three intuitive parameters or channels: hue, saturation and value (or brightness). Roughly speaking, hue allows identifying and distinguishing colours in the same way as the names of colours do by using an angle from 0 to 360 degrees. For instance, red is at 0°, green at 120° and blue at 240°. Alternatively, it can be represented as well as a percentage from 0 to 100% (or 0 to 1). In the three-dimensional HSV encoding, hue is the first channel. Saturation can be thought as a parameter that measures the purity of the colour, i.e. how close is the colour to a black & white tone. It is expressed as a percentage, being 100% fully saturation and 0% a black & white tone. In the three-dimensional HSV encoding, saturation is the second channel. Finally, the third channel, the parameter value, has the same interpretation as the value of the pixels of a black & white image, i.e the intensity. Therefore, HSV model is a straight-forward way of quantitatively represent colour. The HSV Colour Space is usually represented graphically as a cone in which the three channels can be properly identified and interpreted.

In order to represent the simplest way how predominant colours of a video are, we make use the pixel values of the HSV channels in every frame of the videos and extend the features to the video-level by computing the average and the standard deviation, similarly to the procedure presented in Section 4.3 for defining the intensity features. Again, black frames at the beginning and the end of the video are not taken into consideration.

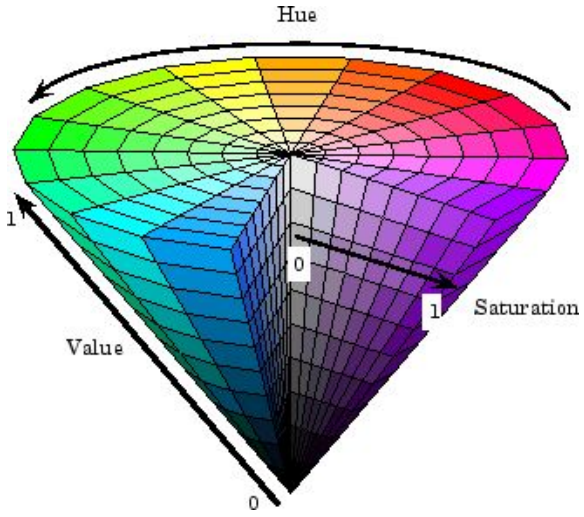


Figure 4.2: Illustration of the HSV Color Space as a conical space. Picture by MathWorks [30]

Average hue

The average hue is the mean value of the first channel of the pixels in the HSV representation of all the frames of a video. This feature, in spite of lying at a very low level, can give an idea of the average tone of the colours of the video.

feature ID	f_{12}
short name	avg-hue
theoretical min	0
theoretical max	1
min	0
max	0.73
mean	0.31
standard deviation	0.12

Table 4.12: Statistics and information of '**average hue**'

Standard deviation of the hue

The standard deviation of the hue along all the frames of a video is a good indicator of monotony of the colours in the video. If the value of this feature is very low, it will mean that very similar colours are used throughout the whole video and vice versa.

feature ID	f_{13}
short name	std-hue
theoretical min	0
theoretical max	1
min	0
max	0.31
mean	0.12
standard deviation	0.05

Table 4.13: Statistics and information of '**standard deviation of the hue**'

Average saturation

The average saturation is the mean value of the second channel of the pixels in the HSV representation of all the frames of a video. This feature is indicative of the average saturation of the colours of the video. A high value indicates that colours in the video are quite bright and saturated on average, whereas a low value means that colours in the video are closer to black & white tones.

feature ID	f_{14}
short name	avg-saturation
theoretical min	0
theoretical max	1
min	0
max	0.59
mean	0.25
standard deviation	0.11

Table 4.14: Statistics and information of '**average saturation**'

Standard deviation of the saturation

Similarly to f_{13} (std-hue), the standard deviation of the saturation represents the variation of saturation on colours along the video. Normally, saturation is not a colour characteristic that varies significantly along the same piece of video, but it might be modified for any reason and this feature should capture this effect.

feature ID	f_{15}
short name	std-saturation
theoretical min	0
theoretical max	1
min	0
max	0.31
mean	0.12
standard deviation	0.06

Table 4.15: Statistics and information of '**standard deviation of the saturation**'

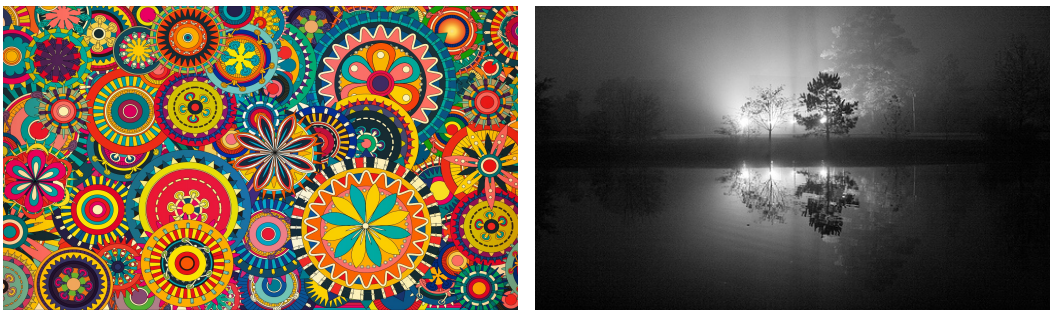
4.5.2 Colourfulness

A colourful picture is referred to as a picture with richly varied colours. From the point of view of analysing advertising videos, it can be interesting to measure the degree of colourfulness of the video in order to learn if the extensive use of colours in the shots can attract people or, on the contrary, the absence of colour can be a characteristic of an attractive video in some cases. Note that in this case, it is not desired to measure the intensity or vividness of colours, which is something that is better measured by features such as the intensity or the saturation. By colourfulness, it is intended to give an idea of the degree of utilisation of a great variety of colours, in contrast to monochromatic or poorly coloured images.

In order to be able to express this idea numerically, we depart from the method presented by [9]. First of all, since this feature is critically related to colour, it is a good practise to perform the image analysis in a colour space that approximates better the human visual perception. For this purpose, before any other operation, each frame is converted into the CIE 1976 L*a*b [36], or simply Lab, colour space, which makes the luminance scale more perceptually uniform.

The idea for measuring the colourfulness of an image, as presented in [9], is to compare the colour distribution of the frame with the distribution of an ideally multi-coloured image. In order to make this comparison, using simply the raw pixel values would be computationally very costly and very noisy as well, due to the three-dimensional nature of a colour image. A smarter approach is to use a discretised colour histogram computed as follows: first of all, each of the three colour channels is divided into four partitions or regions, giving rise to 64 possible combinations or blocks in total, i.e. 64 different colours, instead of the $256^3 = 16,777,216$ possible colours with an 8-bit coding. Then, each pixel in a frame is classified into one of the blocks, building all together a distribution or histogram with 64 bins. In order to measure how colourful the frame is, it is compared to the ideally multi-coloured image, which is assumed to be an image whose colour histogram follows a uniform distribution with values $\frac{1}{64}$.

The technique used for comparing both distributions is the Earth Movers Distance (EMD) [46], a measure of similarity between two distributions that represents the minimal cost that must be paid to transform one distribution into another. It can be efficiently computed through algorithms of linear programming. It is important to remark here that, since the EMD measures the minimum cost to transform the colour histogram distribution of a picture into the distribution of an ideally multi-coloured image, the more colourful a frame is, the lower the value of the feature will be. We provide a couple of examples of pictures and their value of colourfulness in Figure 4.3



(a) $C = 63.432$, [16] (b) $C = 122.19$

Figure 4.3: A multi-colour and a black & white picture with their values of colourfulness

The picture on the left has many different colours and thus a little distance to an ideally multi-coloured image and low value of the colourfulness feature, while the picture on the right is a black & white picture and the feature value is greater.

As it has been done with other features, this has been extended to the video-level by taking the average colourfulness along all the frames of the videos and computing the standard deviation of the distribution, without taking into consideration the black frames at the beginning or the end of the video. To better illustrate the procedure for computing the colourfulness, a summary of the algorithm to obtain these features is shown in Algorithm 4 in Appendix B.

Average colourfulness

The average colourfulness along all the frames of a video gives an idea of the richness of colours in the video. As it has been stated before, a colourful video should have a low value of this feature. It is important to remark again, that in this case we are not measuring the saturation, purity, intensity or hue of colours, but the colour variety or richness of the video.

feature ID	f_{16}
short name	avg-colourfulness
theoretical min	$\tilde{58}$
theoretical max	$\tilde{154}$
min	82.24
max	145.89
mean	105.75
standard deviation	11.37

Table 4.16: Statistics and information of '**average colourfulness**'

Standard deviation of the colourfulness

This feature is an indicator of how the distribution of frame colourfulness of the video is. For instance, a video that alternates colourful shots with black & white shots should have a high value of standard deviation of the colourfulness.

feature ID	f_{17}
short name	std-colourfulness
theoretical min	0
theoretical max	$108.89 = \frac{1}{\sqrt{2}} \times 154$
min	2.47
max	29.23
mean	16.96
standard deviation	5.04

Table 4.17: Statistics and information of '**standard deviation of the colourfulness**'

4.6 Rule of Thirds

The rule of thirds (RoT) is one of the most important rules of thumb in visual arts, such as photography, painting or design. Applied to pictures, it is a rule of composition that states that the most important subjects in the image should be placed either at the horizontal and vertical imaginary lines that divide the image in thirds, giving rise to nine equal parts, or at the intersection of these lines.

The idea behind using thirds is that it divides the frame into pieces whose proportions approximate the golden ratio, widely present in nature and used already by ancient Greeks in architecture, sculpture and other arts because it gives harmony to the compositions. Apart from being a guide to place the subjects, this rule is also followed to place the line of the horizon or any other horizontal dividing line in the image. If it is placed at the lower third line, it will give more strength and priority to the sky or the upper part and, analogously, if it is placed at the upper line, it will give more strength and priority to the ground or the lower part. In video filming this rule is also widely followed for placing moving subjects and the horizon, specially when filming landscapes. On account of this, we have developed a technique for measuring the degree of utilisation of the rule of thirds for placing the horizon or the important horizontal lines.

The technique we propose is based on measuring the difference between the lower and the upper parts of the image divided by the approximated imaginary horizontal line, because it can be observed that when the RoT is followed for this purpose, there is usually a significant difference between both parts of the picture. However, using simply, for instance, a difference of the average intensity would not provide good results. A richer parameter, such as colour, should be used.

Figure 4.4: A sample image with the horizontal and vertical third lines. Picture by Trey Ratcliff [43]

Colour provides enough information to be able to measure the difference between the two parts of an image. Since the three-dimensional nature of colour representation is not very easy to handle, it was decided to use a discretised colour histogram computed equally to how it is done for computing the colourfulness, as presented in Section 4.5.2: each colour channel is partitioned into four regions, giving rise to 64 possible combinations or blocks, i.e. 64 different colours. Then each pixel of the picture (or, in this case, part of a frame) is assigned one of these blocks and two distributions or colour histograms with 64 bins each are obtained. Finally, once the colour histograms of the two regions of the frame are computed, they must be compared. In order to do this comparison, we use the sum of absolute differences (SAD) of the histograms as a measure of the degree of utilisation of the RoT applied to place the horizon or any other dividing horizontal line:

$$D_{ROT} = 32 \cdot \frac{1}{64 \cdot H \cdot W} \sum_{b=1}^{64} |H_{top}(b) - H_{bottom}(b)| \tag{4.6}$$

The value of the measure is higher when the difference of the histograms is bigger, hence the higher the value of this parameter, the higher the degree of utilisation of the RoT should be, as can be seen in the images in Figure 4.5, from which the value of the parameter applied to the lower third line has been calculated.

(a) $D_{ROT-L} = 0.930$, [51] (b) $D_{ROT-L} = 0.338$, [42]

Figure 4.5: Examples of pictures on which the rule of thirds have and do not have been followed, with the values of the measure D_{ROT-L}

One problem with using this method is the fact that if the RoT is followed on one of the horizontal thirds lines (upper or lower), it will make the parameter that measures the degree of

utilisation of the RoT at the other line take a relatively high value, which is not desirable. Therefore, we define some *penalties* in order to avoid this side effect: if the value of the measure for one of the lines is higher than 0.7 (a value near, but slightly lower, to the average of the measure for frames at which the RoT has been followed in some way) and the value of the parameter for the other line is greater, the value of the first is halved. This penalty sets a value in the range of those that do not follow the RoT in these particular cases.

Similarly, we define another penalty to correct the results when the rule of thirds has not been followed, but an important horizontal line has been placed in the middle of the frame, dividing the image in two approximately equal parts with significantly high differences, as is the case in the image at Figure 4.6.

Figure 4.6: An image in which an important horizontal line in the middle [44]

In such a case, the value of the measure for both the upper and the lower third lines might be relatively high, even though the RoT has not been followed at all. Actually, placing the dividing line in the middle of the picture is considered in photography as a violation of the rule of thirds and therefore, the value of the measure should be small for both the bottom and the top lines. To handle this situation, we make use of the fact that if a similar measure is applied to compare the two halves, a high value is obtained, and another penalty should be applied in these cases: if the ratios between the value of the parameter for each of the thirds lines and the value for the middle line are less than or equal to 0.95, the value of each of the parameters is multiplied by 0.33. In this case, we use a ratio and a threshold instead of a direct comparison in order to set a security range for avoiding penalising actual cases of utilisation of the rule of thirds in which the value of the parameter for the middle line is high for any other reason.

In order to illustrate all these procedures and penalties, a summary of the algorithm that computes the parameters of degree of utilisation of the RoT on a single frame is shown in Algorithm 5 in Appendix B. We extend the value of these measures to the video-level by taking the average along the whole video and computing the standard deviation of the distribution, as usual.

Average degree of utilisation of the RoT at the lower third line

This feature is the average value of the previously defined descriptor along all the frames of a video, applied to the comparison between the sub-images below and above the lower third line, without considering the black frames. A high value of this feature might be indicative of a preference for using the RoT in the video and, particularly, if applied to landscapes, for giving more relevance to the sky, for example.

feature ID	f_{18}
short name	avg-hrot-lt
theoretical min	0
theoretical max	1
min	0.04
max	0.59
mean	0.27
standard deviation	0.09

Table 4.18: Statistics and information of '**degree of utilisation of the rule of thirds on the lower third line**'

Standard deviation of the degree of utilisation of RoT at the lower third line

The standard deviation of the degree of utilisation of the RoT on the lower third might be useful to determine if videos are less or more consistent on following the rule of thirds.

feature ID	f_{19}
short name	std-hrot-lt
theoretical min	0
theoretical max	$\frac{1}{\sqrt{2}}$
min	0.03
max	0.28
mean	0.16
standard deviation	0.05

Table 4.19: Statistics and information of '**standard deviation of the degree of utilisation of the rule of thirds on the lower third line**'

Average degree of utilisation of the RoT at the upper third line

Similarly to f_{18} (avg-hrot-lt), this feature is the average value of the previously described feature along all the frames of a video, applied to the comparison between the sub-images below and above the upper third line, without considering the black frames. In this case, a high value of this feature might be indicative of a preference for using the RoT in the video and for giving more relevance to the ground than to the sky.

feature ID	f_{20}
short name	avg-hrot-ut
theoretical min	0
theoretical max	1
min	0.04
max	0.59
mean	0.27
standard deviation	0.09

Table 4.20: Statistics and information of '**degree of utilisation of the rule of thirds on the upper third line**'

Standard deviation of the degree of utilisation of the RoT at the upper third line

The interpretation of this feature is identical to the one of f_{19}, but applied to the upper third line.

feature ID	f_{21}
short name	std-hrot-ut
theoretical min	0
theoretical max	$\frac{1}{\sqrt{2}}$
min	0.02
max	0.30
mean	0.16
standard deviation	0.06

Table 4.21: Statistics and information of '**standard deviation of the degree of utilisation of the rule of thirds on the upper third line**'

4.7 Technical Aspects

In the previous sections we have presented the 21 features which have been used for the classification experiments with the aim of studying their correlation with the perception of the videos from users. In order to implement the algorithms that extract these features from the frame decomposition of the videos we have turned to MATLAB, which offers a set of powerful functions and tools to handle images, particularly with the *Image Processing Toolbox*. We have implemented a MATLAB function to read the videos and extract the basic properties, such as the duration or the frame rate, and we have written several functions for extracting the different features related to each visual aspects. Not only the above mentioned parameters were extracted and stored from each video, but also other characteristics were computed for each video. These characteristics are not useful for the classification purposes, but they are of great interest for analysing the properties of the features and improve their applicability. For instance, the frame distributions of every feature were stored as vectors for further analysis. All these functions are brought together in a bigger script that extracts all the features and distributions from a given video, post-processes the information and stores the results in files.

The computational cost of performing all the required operations of image processing for extracting all the descriptors from every video is significantly high and nearly infeasible by a regular computer or laptop within a reasonable amount of time. It is important to remind that the features are computed by performing operations on every single frame of each video, something that requires an enormous computational effort, taking into account that operations such as extracting the colourfulness need several seconds, as it has to transform the colour space, compute a colour histogram of the frame and solve a linear programming problem in order to get the values. As a way of illustration, the *durations* of the 315 videos from the corpus sum up to 13,428 seconds, which on an average of 25 fps, yield to more than 335,000 frames from which to compute feature values. Facing this situation, it was decided to make use of the computer cluster of the Departamento de Teoría de la Señal y Comunicaciones of the Universidad Carlos III de Madrid, which, apart from having more computational power, can extract features from several videos in a concurrent manner.

Chapter 5

Classification Experiments

Once acquired the data sets with proper annotations and obtained the values of the set of features for every video, the last step of the experimentation process is to carry out the classification experiments that will provide the results to check if the objective of demonstrating that it is possible to learn subjective information related to the appeal of car advertising videos from visual features is fulfilled. However, this step does not consist only in applying a classification algorithm to the data set to obtain an accuracy percentage. Following the criteria applied to the previous steps, we decided to design and implement a system that enabled us to explore different alternatives and handle the great amount of input data sets.

The whole classification process that has been implemented consists of three main steps: a feature selection procedure to conveniently reduce the number of visual features to be involved in the classification experiments, the application of classification algorithms to evaluate how well the selected features can predict the polarity labels and a final step of automatic analysis of results to obtain a comparison of the results and information of statistical relevance. Along the subsequent sections of this chapter, these three main steps will be explained in more detail. At the end of the chapter the scripts and procedures that bring all the pieces together will be discussed.

5.1 Feature Selection

Feature selection methods have been added to the machine learning process of this research project to improve the efficiency and performance of the classification. Feature selection is often a very good practise before performing classification. Along this section, we will provide an introductory explanation of the basic aspects of the topic and then we will stick to the relevant aspects of our work, the functionality of the feature selector provided by WEKA and the algorithms we have used. Let us recall that information about WEKA is provided in Appendix A.

5.1.1 Introduction and Basic Aspects

Feature selection, also known as attribute selection, is a tool or technique from machine learning whose aim is to reduce the dimensionality of data sets by removing some features under redundancy or irrelevance criteria. Feature selection is used in machine learning applications for three main purposes:

- **Improving the prediction performance** of classification algorithms: having a very high number of features, rather than helping classifiers make better predictions, can affect negatively to the generalisation of results due to overfitting problems.

- **Reducing the complexity** and computational cost of classifiers: although it is not the main issue of the classification process, high-dimensional data set might eventually turn the algorithms into very costly processes.
- **Providing an easier comprehension and interpretation of the results** and of the underlying process that generated the data: the fewer variables a problem has, the easier to understand it should be.

There exist two broad categories into which most feature selection methods typically fall: *wrappers* and *filters*. The former make use of the learning algorithm to be applied to the data for evaluating the worth of features, while the latter evaluate the worth of features by using the general characteristics of the data. Depending on the scenario, wrappers might be more convenient than filters, but in general filters work faster and offer more general results.

Independently on the type of feature selection method, they all can be defined as a combination of a **search technique** to identify and propose new feature subsets, together with an **evaluation measure** which scores the worth of the different subsets. More concretely, any feature selection method follows the next main steps, according to [3]:

1. **Starting point**: it involves determining the point at which the search starts. There are several alternatives: starting with no features and progressively add the features, starting with all features and progressively remove the features or starting at some point in the middle. These methods are called *forward selection*, *backward elimination* and *outward progression*, respectively.
2. **Search organisation**: considering all possible subsets with N features yields 2^N combinations, which is not an option to consider for large data sets. Instead of this exhaustive search, heuristic search strategies can be used to provide good results more efficiently, although they might not provide the optimal set.
3. **Evaluation of subsets**: at some point, there must be a method for evaluating the subsets and compare them. The selection of the strategy for performing this task can affect heavily the results of the algorithm. As it has been commented before, there are strategies based on a particular learning algorithm to test the usefulness of the subsets by induction, called wrappers, and strategies based on intrinsic data information, such as correlation coefficients or mutual information, called filters.
4. **Stopping point**: there are different criteria to decide when to stop the algorithm depending on the evaluation strategy and the starting point. The algorithm can continue adding or removing features until no subset improves the performance, until the variation does not increase the merit of the subset or until the whole space has been covered.

5.1.2 WEKA Attribute Selection Algorithms

As for many of the operations related to machine learning that have been done in this project, the feature selection procedures have been performed with WEKA [14]. This machine learning software provides a functionality for applying attribute selection techniques on a given data set. There are available 17 attribute evaluators, both wrappers and filters, which can be combined, depending on their nature, with 11 search techniques. Furthermore, WEKA allows performing the attribute selection either on the full training set or with a K-fold cross-validation algorithm. For our purposes, the feature selection procedures have been done with 10-fold cross-validation and from the available combinations of attribute evaluators and search methods, the 6 following algorithms (*attribute evaluator / search method*) have been selected:

1. **CFS Subset Evaluator (CfsSubsetEval) / Best First**: CFS stands for correlation-based feature selection, so CfsSubsetEval is an attribute evaluator which uses the correlation between features to find redundancy and eliminate features. Therefore, it is a filter feature selector. According to Hamilton in [13], who developed the algorithm, the functionality of this

method is based on the hypothesis that "a good feature subset is one that contains features highly correlated with the class, yet uncorrelated with each other". The Best First search method, with the configuration we used, starts with an empty set and searches the space of attribute subsets by greedy hill-climbing augmented with a backtracking facility. This attribute selection method can provide subsets of 1 or 2 features.

2. **SVM Attribute Evaluator (SVMAttributeEval) / Ranker**: this is a wrapper method which evaluates the worth of an attribute by using a support vector machine, by ranking the attributes with the square of the weight assigned by the SVM. This evaluator must be used together with a Ranker search method, which simply uses the individual evaluations to create the subset with the N best features. We have run this attribute selection algorithm by setting the value of N from 1 to 10. Therefore, through this method, from a single data set, we generate 10 different feature subsets.

3. **Consistency Subset Evaluator (ConsistencySubsetEval) / Greedy Stepwise**: ConsistencySubsetEval is a filter method which evaluates the worth of a subset by the level of consistency in the class values when the training instances are projected onto the subset of attributes. This method only considers subsets with consistency equal to that of the full set of attributes, so it is used in combination with a greedy search, which performs an exhaustive forward search of subsets of size N from 1 to 10. With this method 10 different feature subsets are generated as well.

4. **Information Gain Attribute Evaluator (InfoGainAttributeEval) / Ranker**: it is a filter method which evaluates the worth of an attribute by measuring the information gain with respect to the class, according to the following definition of information gain:

$$InfoGain(Class, Attribute) = H(Class) - H(Class|Attribute)$$

where H is the information entropy. InfoGainAttributeEval has been used together with the Ranker search method, to create 10 features subsets by running the algorithm with values of N from 1 to 10.

5. **Principal Components Evaluator (PrincipalComponents) / Ranker**: this evaluator performs a principal components analysis, which consists in applying orthogonal transformations to the features to convert them into a set of linearly uncorrelated combinations. It is used together with a Ranker method to search for N-dimensional subsets, from 1 to 10. The main drawback of this method is that the features from the subsets are no longer the original features, but linear combinations of them. Thus the interpretation of the features becomes very complicated. However, we add this method to our analysis to test if such a particular procedure could yield good classification results on our data sets.

6. **Classifier Subset Evaluator (ClassifierSubsetEval) / Race Search**: this wrapper method provided by WEKA allows selecting any classifier to evaluate the merit of the subsets selected by the search method. In our case, the classifier we chose is a KStar algorithm, which had provided good classification results in some preliminary experiments. The search method selected for this feature selection procedure is a Race Search. This algorithm races the cross-validation error of competing attribute subsets, built through a forward selection method, starting with an empty set and letting the subset grow until there is no further improvement. Therefore, in this case only one feature subset is generated.

5.1.3 Implementation

Due the amount of data sets generated in the clustering step and the large number of experiments required at this feature selection step, we needed to implement an automatic procedure. For this purpose we took the advantage that WEKA, apart from a graphical user interface, provides some functionality that can be executed from a command line interpreter. This is the case of the attribute

selection techniques and therefore we wrote a shell script (.sh) which receives as inputs the data set on which the feature selection has to be performed, the output file to store the feature subset and the strings with the specification of the attribute evaluator and the search method together with their corresponding configurations.

It is important to recall that the inputs of this module are the 84 data sets generated in the cluster analysis (see Table 3.6). These data sets were arranged as ARFF files (see A.1) containing 138 or 315 instances, depending on the list, and the values for each video of the 21 visual features. Moreover and, very importantly, each instance has a field with the identifier of the class it belongs to, assigned in the clustering process. Regarding the outputs of this module, they will be ARFF files as well, but instead of 21 feature values, each instance will have a smaller number of them. It has been explained in the previous section that 6 different attribute selection methods are applied to the input data sets, and that each of these methods generates one or more feature subsets. In order to be more precise with the number of outputs of this module, we present in Table 5.1 the characteristics of the feature subsets that each attribute selection method generate.

Attribute Selection Method	**# Subsets**	**# Features in Subsets**
CfsSubsetEval \ BestFirst	1	1 or 2
SVMAttributeEval \ Ranker	10	1–10
ConsistencySubsetEval \ GreedyStepwise	10	1–10
InfoGainAttributeEval \ Ranker	10	1–10
PrincipalComponents \ Ranker	12	1–12
ClassifierSubsetEval \ Ranker	1	indefinite
	44	

Table 5.1: Number of feature subsets generated by each attribute selection method and number of features in these subsets

The whole process of feature selection involving all the operations required for executing the 44 different configurations of the algorithms has been implemented in a MATLAB module, which iterates over the number of possible configurations for each algorithm and executes the shell script with the proper inputs. A summary of the operations performed in this module on a single data set can be seen in Algorithm 6 in Appendix B for a better illustration of the procedures.

At the end of the feature selection module, a total of $84 \times 44 = 3,696$ feature subsets are to be created and will be the input of the next module, the classification through several machine learning algorithms.

5.2 Classification

Although we refer to classification, this important step of the project refers to the broader process of machine learning, which consists in first, training a system according to input variables and second, testing the trained model in a classification test of unseen samples. In this research work, it is intended to train the classifiers through the visual feature values, together with the labels assigned in the cluster analysis. In this section, we will start offering a brief introduction to the main aspects of machine learning and classification, taking special notice to those relevant to this work. Then, we will explain some details of the possibilities that WEKA offers regarding classification and will introduce the main aspects of the classifiers used in this work. Finally, some details of the implementation of this module will be given.

5.2.1 Introduction and Basic Aspects

Machine learning is a field of artificial intelligence which concerns the study of some input data with the intention of learning from them, extracting certain knowledge and generalising this knowledge to unseen data instances. According to Arthur Samuel, a pioneer in the field of artificial intelligence, machine learning "gives computers the ability to learn without being explicitly programmed". This powerful ability is behind many successful applications and developments, such as speech recognition, fingerprint identification or DNA sequence recognition and, as it is the aim of this work, it can also be efficiently applied in conjunction with video processing techniques in the field of computer vision.

The Learning Process

The learning process, illustrated in Figure 5.1 takes some known input $\mathbf{x}$ and output $\mathbf{y}$ data[1] and focuses on the prediction and statement of a hypothesis $g : \mathcal{X} \to \mathcal{Y}$, generalisable to any unseen data with some error $\mathcal{E}$, basing on properties learnt from the input data. This process departs from the assumption of the existence of an unknown target distribution $P(y|\mathbf{x})$, which represents an unknown target function $f : \mathcal{X} \to \mathcal{Y}$ plus some noise. Learning is achieved through learning algorithms $\mathcal{A}$ that make use of a hypothesis set $\mathcal{H}$, from which the final hypothesis g is extracted.

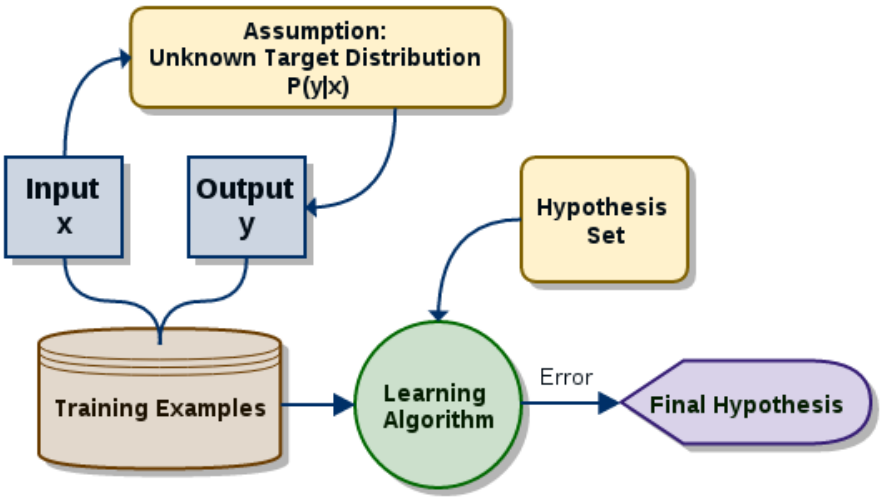


Figure 5.1: Diagram showing a summary of the learning process

Depending on the nature of the target function, the learning algorithm will perform either regression or classification. Regression concerns real-valued outputs $\mathcal{Y}$, while in classification problems each input x is assigned a category y. Although regression techniques are usually applied to classification problems, since the extent of this work is the latter, we will refer to classification as the final objective. If we relate the peculiarity of classification to the learning process, we can say that classification is the task of identifying which category a new observation belongs to, using the knowledge acquired and learnt from the training data. Regarding the learning algorithms, if applied to perform classification, they are called classifiers. There exists a wide variety of classifiers, from which one of the most popular families are linear classifiers, decision trees, support vector machines, Bayesian networks and neural networks. The key aspects of the classifiers used in this research work will be presented in the next section.

[1] In the case of supervised learning, the output y is really known information from the beginning. However, in unsupervised learning, as it is the case of this work, this output y must be also learnt in advance. For simplicity, in this explanatory section y is assumed to be already known

Validation

There is an important aspect involved in the learning process and thus, in classification problems referred to as validation. Given a data set $\mathcal{D}$ with N instances, it can be divided before starting the learning process into two separate subsets $\mathcal{D}_{train}$, with $N - K$ instances, and $\mathcal{D}_{val}$, with K instances, with the aim of helping the algorithm make learning choices, such as when to stop the iterations of a classifier (early-stopping). However, there is a critical trade-off in the choice of K, the size of the validation set, because the bigger K is, the more reliable the validation will be, but also the worse the estimate might be, because there will be fewer instances for training, and vice versa. In order to minimise the effect of this trade-off, there is a solution known as K-fold cross-validation, consisting in performing $\frac{N}{K}$ training sessions on $N - K$ different instances and average the results of all sessions. The error of the estimate using cross-validation can be defined as follows:

$$E_{CV} = \frac{1}{N} \sum_{n=1}^{N} e_n$$

5.2.2 WEKA Classifiers and Experimenter

Similarly to the Attribute Selection, WEKA provides functionality for performing classification techniques on data sets. It includes more than a hundred classifiers and provides a very complete set of statistical measures as results, such as accuracy, parameters related to the confusion matrix or the F-measure, among others. Furthermore it also allows performing K-fold cross-validation on the classification experiments. Apart from the basic functionality of testing a classifier on a single data set, WEKA provides a powerful tool called Experimenter, which allows testing a set of previously defined classifiers on several data sets, including not only the cross-validation technique, but it is also possible to perform a number of random iterations to test the experiment under different situations and increase the reliability of the results. The output of the Experimenter is an ARFF file containing all the information and results from every simulation, with proper attributes for identifying the data set, classifier, fold and iteration to which each line belongs.

We have configured the experimenter so that it performs on each feature subset 10 random iterations of 10-fold cross-validation for each of the 13 different classification algorithms we have selected. We have tried to select classifiers from different families so that we can test the performance of distinct types of learning algorithms. Next, a brief explanation of the main aspects of these classifiers is presented:

Rule-based classifiers

- **ZeroR**: the Zero Rule algorithm can be seen as the simplest classifier because it relies only on the classes, ignoring the features. It simply predicts the mode of the classes with an accuracy equal to the relative frequency of the most frequent class. Although it is useless for learning purposes because it does not really predict anything, it is normally used for determining a benchmark for the rest of classifiers. Therefore, ZeroR will be our baseline scheme for analysing the results in the next chapter.
- **OneR**: this is another simple algorithm, but it achieves reasonably good performance. It computes the frequency of the classes for all the values of each feature, discretising first numeric values, computes the total error committed by the features if the most frequent class for each value is selected and finally selects the feature that minimises the total error as its unique rule.

Bayesian classifiers

- **BayesNet**: the Bayes Network classifier makes use of a probabilistic graphical model that represents the set of features with their conditional dependencies via a directed acyclic graph, whose nodes are assigned a probability function that can be represented with conditional probability tables. In the configuration we have selected for the WEKA implementation of this classifier, we have chosen a Simple Estimator for computing the probability tables of the features, which estimates probabilities directly from data, and a K2 search algorithm, which uses hill-climbing for selecting the variables and building the network structure.

- **NaiveBayes**: this classifier builds a much simpler network by assuming independence between features, conversely to Bayes Networks. Furthermore, it uses a simple way of computing the posterior probabilities of each class and features, using the Bayes' theorem:

$$P(c|f) = \frac{P(f|c)P(c)}{P(f)}$$

  where c is the class, f is the feature, $P(f|c)$ is the likelihood and $P(c|f)$ is the target probability of each class, given a feature. In spite of its simplicity, this classifier often achieves better performance than more sophisticated algorithms.

Function-based classifiers

- **SimpleLogistic**: this classifier is a logistic regression model. It is based on a linear classifier, which have typically the form $s = \sum_{i=0}^{d} w_i f_i$, but while a linear classifier uses a line for determining the class of an instance, this classifier uses a logistic curve that can be defined with the logistic function $\theta(s) = \frac{e^s}{1+e^s}$. The logistic regression finds the weights w_i that minimise the error through an iterative method called gradient descent because it finds the minimum by traversing the gradient of the error:

$$\nabla E = -\frac{1}{N} \sum_{n=1}^{N} \frac{c_n \mathbf{f}_n}{1 + e^{c_n \mathbf{w}^T \mathbf{f}_n}}$$

  The implementation of this classifier in WEKA is based on the LogitBoost [49] learner and the works of [24], containing certain particularities with respect to a regular implementation of the logistic classifier.

- **Logistic**: this implementation of a logistic classifier that WEKA includes is a generalisation of the logistic regression specially designed for multi-class problems and it intends to be an improvement of SimpleLogistic as it incorporates a ridge estimator, implemented as a variation of the algorithm presented by le Cessie and Houwelingen in [25]. This additional feature improves the performance of the logistic regression when the number of covariates is particularly large or they are largely correlated.

- **SMO**: this classifier trains a support vector machine (SVM) using the sequential minimal optimisation (SMO) introduced by John Platt [40] and improved later in [20]. The basis of SVM is a linear classifier in which it is intended to optimally separate the classes by maximising the margin between instances from different classes. As any linear classifier, the aim of an SVM is to find the weights of the features. In this case, the idea for finding the optimal weights consists in maximising the distance between any instance x_n and the hyperplane that separates the classes:

$$maximise \frac{1}{\|w\|}$$
$$subject\ to \min_{n=1,2,\cdots,N} |\mathbf{w}^T \mathbf{x}_n + b| = 1$$

At the end of the optimisation, the closest instances to the separating plane are called support vectors and are said to achieve and support the margin. Multi-class problems can be solved with the WEKA implementation using pairwise classification, according to the design of [15].

- **MultilayerPerceptron**: this classifier is a biologically inspired algorithm referred to as *feedforward artificial neural network*. A multilayer perceptron is a combination of multiple single perceptrons, which is the simplest linear classifier. The perceptrons are organised in layers (l) of nodes or neurons (x) which are fully connected to the next one. The neurons of the first layer represent the features of the data set, while the inner layers are hidden and thus, unknown. The connections of the neurons are the weights (w) of the perceptrons that must be computed. The value of each neuron is related to the neurons of the previous layer as follows:

$$x_j^{(l)} = \theta \left(\sum_{i=0}^{d^{(l-1)}} w_{ij}^{(l)} w_i^{(l-1)} \right)$$

where i refers to the inputs, j to the outputs and θ is the activation function, a sigmoid defined as:

$$\theta(s) = tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

The weights are computed following an algorithm called *back propagation*, a generalisation of the least mean squares algorithm that iterates over the layers starting at the last one and computing the weights backwards.

Tree-based classifiers

- **J48**: this classifier implements the C4.5 algorithm, developed by Quinlan [41]. This algorithm builds a decision tree, whose nodes are built by choosing the feature of the data that most effectively splits its set of samples into subsets enriched in one class or the other, using the information entropy as the splitting criterion.
- **ADTree**: this classifier is an alternating decision tree algorithm, developed by Freund [12] in 1999 and it is a generalisation of decision trees. An ADTree consists of decision nodes and prediction nodes. The former specify a predicate condition and the latter only contain a single number. Instances are classified following all paths for which all decision nodes are true and summing any prediction node that is traversed. The main drawback of this classifier is that its WEKA implementation is designed only for two-class problems and therefore, we cannot obtain results with this classifier for the cases with more than two classes.
- **RandomTree**: this algorithm builds the tree by randomly choosing K features at each node. The decision is taken at leaf nodes, which output a class probability distribution. In the default implementation of this classifier in WEKA, the value of K is

$$K = \log_2(\#features) + 1$$

- **RandomForest**: this classifier is an algorithm that combines multiple random trees developed by Breiman in [5]. Each random decision tree is trained with a subset of instances and the final decision is the class that is the mode of the classes output by individual trees.

Instance-based classifiers

- **KStar**: this classifier is an instance-based classifier developed by Cleary in 1995 [8]. The main characteristic of this classifier with respect to others of its type, is that it uses a measure of similarity based on an entropy distance, which intuitively represents the complexity of transforming one instance into another.

5.2.3 Implementation

Similarly as for the feature selection module, in order to carry out the classification experiments we have used the command-line implementation of the WEKA Experimenter and built a shell script that executes it, which can be called from a MATLAB script. It is important to recall that the inputs of the experimenter are the 3,696 feature subsets generated in the previous module. For each feature subset, there is an execution of the Experimenter, which will test the performance of the 13 classification algorithms presented in the previous section. For each data subset, the results of every experiment of all the classifiers are stored in an ARFF file. The path of the feature subset is an argument of the shell script that executes the Experimenter and the output path is specified as a tag of an XML configuration file, which also contains the specification of the classifiers with their proper parameters and other configuration parameters of the Experimenter, such as the definition of the 10-fold cross-validation and the 10 random repetitions of each experiment.

The shell script is called from a module of a MATLAB script, which, due to the fact that the details and complexity of the classification process are encapsulated into the configuration file of the Experimenter, is much simpler than the one for the feature selection. In order to illustrate the procedures of this module, Algorithm 7 in Appendix B shows the steps for executing the WEKA Experimenter and run all the required experiments just for one data set from the total of 84.

5.3 Automatic Analysis of Results

The last step of the experimentation process consists of two parts or objectives. On the one hand, organising and compacting the multiple results related to classification provided by the Experimenter so that it is simpler to evaluate them. On the other hand, this step, done with another WEKA tool called Analyser, provides an analysis of statistical significance, which is of major importance to discern which results are useful and generalisable and which not.

Along this section, a brief introduction to the notion of statistical significance will be offered in order to better understand the importance of this concept regarding the evaluation of results. Then, we will look at some details of our implementation of this module and the characteristics of the tool provided by WEKA.

5.3.1 Introduction to Statistical Significance

Statistical significance is a very important concept within the field of statistical data analysis. This discipline offers researchers tools and ways to check if there is enough evidence to prove their hypotheses. For example, it provides measures to conclude that two group means are different, that two distributions are different or that there is a linear association between two variables. When there is statistical evidence to prove these situations, it is said that there exists statistically

significant difference. Statistical significance is often proved through tests that depart from the so-called null hypothesis, H_0, that there is no difference between the means or distributions of study. These tests, called hypothesis testing, are designed to determine whether evidence rejects the null hypothesis with a certain level of significance.

The notion of significance implies that some errors might occur. These are denoted as Type I error and Type II error. The first one is the error made when the null hypothesis is rejected, but it is actually true. This error is expressed by α and is often set to 0.05, meaning that there would be an error 5% of the time. The Type II error occurs when the null hypothesis is not rejected, but it is actually false. It is called β and is an important parameter because $1 - \beta$ measures the power of the test, which quantifies the chance of rejecting H_0 when it is false. Thus, the higher the power, the better the chance of finding a difference when it in fact exists. These errors are heavily affected by the sample size. Particularly, the power can be increased by increasing the sample size. Another important parameter to understand hypothesis testing and statistical significance is the p-value, which denotes "the probability of obtaining results as extreme or more extreme than the ones observed given that the null hypothesis is true", according to [11]. Therefore, the smaller the p-value, the more evidence to reject the null hypothesis. Normally, a p-value lower than 0.05 indicates that the null hypothesis should be rejected at the $\alpha = 0.05$ level.

Paired t-Test

One of the most important tests in statistical data analysis is the Student's t-test, a procedure introduced by William Gossett in the early 20th century. The most common use of this test is to compare sample means, which is the application that has been used in this work. There are three types of t-tests according to the nature of the samples and the type of comparisons [11]:

1. One-sample t-test: it is used when the purpose is to compare a sample mean to a fixed number, usually known as "gold standard", that is not based on the collected sample. For example, it could be used to determine if the average volume of liquid in soft drink cans matches the specified volume advertised in the label.

2. Two-sample t-test: it is used to determine if the unknown means of two populations are different from each other, based on independent samples collected from the two populations. An example of use could be determining if the reaction to a certain drug is different in females and males, by doing measurements on sample individuals from both groups.

3. Paired t-test: this version of the test is similar to the two-sample t-test, but this particular variation is used when the samples are related or paired in some way. It is typically applied, for instance, on before and after measurements on a single group of subjects.

Depending on the characteristics of the experiment one particular test must be chosen. In this brief introduction to this statistical test, we will focus on the paired t-test because it is the one of application in this work to compare the average accuracy of the classifiers with the baseline classifier. In this case, the sample consists of all the accuracy results of each iteration and each fold of the cross-validation experiment of a classifier. A paired t-test computes the differences between scores of the two samples and uses this difference as a single distribution, X_D, from which some statistical measures such as the mean ($\overline{X}_D$) and the standard deviation (s_D) are computed. Hence, the null hypothesis for a paired t-test is typically that the population mean of differences μ_d is zero ($H_0 : \mu_d = 0$). As any Student's t-test, a t-statistic is computed and, with this value and the degrees of freedom (df) of the sample, the p-value can be derived. For the paired t-test, the t-statistic is computed as follows:

$$t = \frac{\overline{X}_D - \mu_d}{\frac{s_D}{\sqrt{n}}}$$

The implementation of the paired t-test provided by WEKA includes a correction to deal with the dependence between the scores of the samples (corrected paired t-test), since the classification results of each fold of the cross-validation are related to each other.

5.3.2 WEKA Analyser

The Analyser tool provided by WEKA uses as inputs the ARFF files generated in the previous module with the classification results of all the experiments. The first application of this tool is automatically averaging the values of the statistics of all the iterations and folds of the cross-validation to produce single classification results for each feature subset and classifier. At this step we have computed and stored for further analysis the values of average accuracy and standard deviations of the set of iterations. Furthermore and, apart from this practical application, the Analyser compares the results of the simulations of each classifier with the baseline accuracy, in our case the one provided by the ZeroR classifier. This comparison is carried out with a corrected paired t-tester, which determines for each classifier if its result is statistically significant and thus, useful. The significance has been computed by establishing a confidence of 95% ($\alpha = 0.05$). The possible options are that one result is significantly better than the baseline, significantly worse or statistically equal. This designation of the prediction accuracy of each classifier will be used to evaluate the results in terms of generalisation and usefulness. The outputs of the analyser are simple CSV files, which contain an identifier of every classifier, the average percentage of correctly classified instances or accuracy, the standard deviation and one variable to designate the status of the result in terms of statistical significance.

5.3.3 Implementation

This automatic analysis of the results can be executed through a shell script using the command-line implementation of the WEKA Analyser, similarly to the Attribute Selection and the Experimenter. Therefore, we have built a shell script that receives as arguments the path of an ARFF file generated by the experimenter and the output path of the CSV file. This script is called from a MATLAB module, which iterates over all the experimenter outputs. Besides the CSV file, which will be useful to be automatically read by other scripts, a .txt version is generated to be more readable.

For better illustration of the procedure, we show in Algorithm 8 in Appendix B a summary of the operations of the MATLAB module that implements this final step of the experimentation just for one data set from the total of 84.

5.4 Experiment Setup

This section is a explanatory summary of the global operations of the classification process, which, as the previous sections reflect, consists of three main parts or modules: attribute selection, classification experiments and automatic analysis of the results. To better understand the global process as a whole and not getting lost in the details, we will offer a summary of the steps and important aspects of the classification process, which is in turn the scheme of the MATLAB script that implements this part of the research.

It is important to recall that the inputs of this process are 84 ARFF files which contain either 138 or 315 instances that correspond to the advertising videos and 21 values of the visual descriptors plus one polarity label. This different files were created during the cluster analysis and they differ between each other for they have been generated using 3 different metadata strategies, 4 clustering distance measures, 4 different number of classes and 2 types of normalisation. Therefore, the idea of the script that executes the classification experiments on all these data sets is to iterate over all this possibilities to apply the operations of classification on every file.

The first step is the attribute selection, applied on each of these ARFF files, which generates $84 \times 44 = 3,696$ new feature subsets, reduced versions of the original files with from 1 up to 12 features instead of 21, which are selected according to algorithms based on removing redundancy and irrelevance. Then, with the help of the WEKA Experimenter, the performance of 13 classification algorithms is tested on every feature subset. Finally, the Analyser provides a single value of accuracy and standard deviation for each classifier, yielding a total of $3,696 \times 13 = 48,048$ classification results. Furthermore, the Analyser provides a designation of statistical significance for each result. This process is illustrated in Figure 5.2.

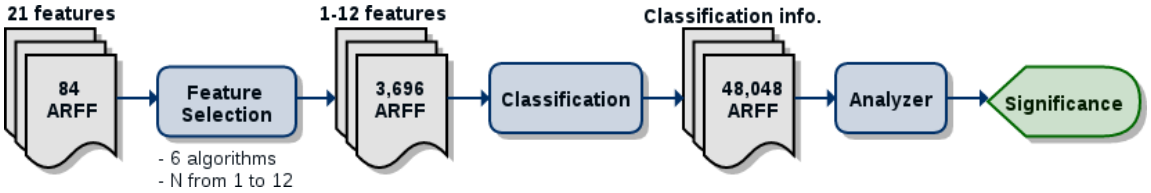


Figure 5.2: Diagram showing the overall machine learning process presented in this chapter

Chapter 6

Experimental Results and Discussion

The aim of this chapter is to make a quantitative and qualitative evaluation of the experimental results that have been obtained during the processes explained in the previous chapters: cluster analysis, extraction of features and classification experiments. On the one hand, we will offer the numerical results yielded by the experiments, paying special attention to percentages of accuracy and statistical significance. On the other hand, we will perform a qualitative analysis of the presented results, with the aim of extracting relevant conclusions.

Due to the high number of layers of analysis which can be extracted from this work, we will organise this chapter in a top-down approach, starting with the broadest issues, such as the comparison between the classification performance of List 1 and List 2, and following towards deeper layers to focus on important details of the analysis and the results. In order not to carry the aspects of top layers towards the deeper analysis and to make a simpler and clearer evaluation, we will discard some elements of the first steps of the analysis as we get into details at further layers. Nevertheless, the distinction between the number of classes generated by the cluster analysis is always kept because classification results on data sets with different number of classes must not be mixed due to their completely dissimilar nature.

6.1 Filtering and Pre-processing of Results

In order to simplify the analysis and to have a cleaner and more precise set of results, before starting the evaluation we first carry out a pre-process that removes some undesired and noisy results. First of all, there are a substantial number of results coming from clustering combinations with more than 2 classes and attempted to be classified with the ADTree classifier. In Section 5.2.2 it was pointed out that the WEKA implementation of this classifier does not support more than two classes. The attempts to classify such combinations with the ADTree are designated with proper identifiers which allow, at this point, to remove them from the set of results to be analysed.

Furthermore, there are 3 additional clustering combinations which lead to an unknown error during the classification process. Therefore, the results coming from this erroneous experiments have been removed from the set of valid results as well. In summary, after this filtering and preparation of the results we have a total of 45,195 valid results, out of the initial 48,048 experiments. Let us recall that this number refers to the total amount of classification results, including those provided by the baseline scheme, the ZeroR classifier. Table 6.1 summarises the valid data sets, feature subsets and experiments which will be taken into account for the evaluation of results.

Data sets	84
Feature subsets	3,693
Classification results	41,502
Classification results (including baseline)	45,195

Table 6.1: Number of valid data sets, feature subsets and classification results

6.2 Best Classification Results

A good first approach to the evaluation of results is to present the best classification results regarding the percentage of correctly classified instances for each number of classes, since we will be in a better condition of starting a deeper analysis of the details if we know from the beginning which is the top performance that have been achieved in this research project. This results are shown in Table 6.2.

	Strat.	**Accuracy** (σ)	**ZeroR** (σ)	**Distance**	**# Feat.**	**Classifier**
2 classes	S_3	72.18 (12.34)	56.48 (1.33)	cosine	4	RandomTree
3 classes	S_1	55.52 (10.94)	47.11 (3.47)	cosine	2	OneR
4 classes	S_3	45.65 (11.19)	35.52 (2.22)	cityblock	2	Logistic
5 classes	S_3	41.71 (4.67)	38.42 (1.06)	sqEuclid.	4	KStar

Table 6.2: Characteristics of the experiments with the best accuracy percentages for each different number of classes

Together with the classification accuracy, some other characteristics of the combinations and experiments that achieved it are shown, in order to better illustrate how the top performances have been reached. However, no strong conclusion should be extracted from these results, since in spite of being the best ones, they might not be representative of the general behaviour of the experiments. In order to explore the general performance of classification algorithms on the set of advertising videos we test, we will go through the details in the next sections, yet keeping in mind the best performance that can be achieved.

6.3 List 1 Vs. List 2

The first distinction that was made in the acquisition process was to split our corpus into two different data sets, as it was explained in Section 3.3.5, based on the different ways of ordering the results provided by the YouTube API. As a reminder, List 1 contains only results provided by *relevance* type of query and List 2, in an attempt of augmenting the corpus, contains results from the three types of queries (*relevance*, *rating* and *viewCount*). It is interesting to compare the performance of both lists for two main reasons. First, because the provenance of some videos of List 2 is quite different from those of List 1 and second, because the size of List 2 is more than double the number of examples in List 1, which is a significant variable in terms of reliability and generalisation of results.

A first and simple way of comparing the performance of both lists is to look at the number of results that improved the baseline scheme with statistical significance. Table 6.3 shows the number and percentage of statistically significant results for each list and we can observe that List 1 has obtained approximately 70% more relevant results than List 2, even though the fact of having more instances should be a factor favouring better significance. This factor is reflected on the fact that for 2 classes, List 2 gets some more significant results than List 1, although as it will shown next, the performance of List 2 is generally much lower.

Another way of comparing both lists is to look at the percentage of correctly classified instances of the experiments, i.e. the accuracy. In this case, in order to get a broader impression of the

	2 classes	3 classes	4 classes	5 classes	Total
List 1	179	44	96	99	**418**
List 2	192	1	44	10	**247**

Table 6.3: Number of statistically significant results for List 1 and List 2

results, not only the significant results will be taken into consideration, but the whole set of valid results. An interesting and descriptive way of representing in the same figure the accuracies of every experiment is using a box plot, which can effectively represent distributions of values. The edges of the boxes are the 25th and 75th percentiles and the whiskers extend towards the most extreme data points not considered outliers, which in this configuration are those points with values further than, approximately, +\- 2.7σ. Outliers have been removed from the plot to simplify the interpretation of the plot. Figure 6.1 shows the distribution of accuracies for each list and number of classes.

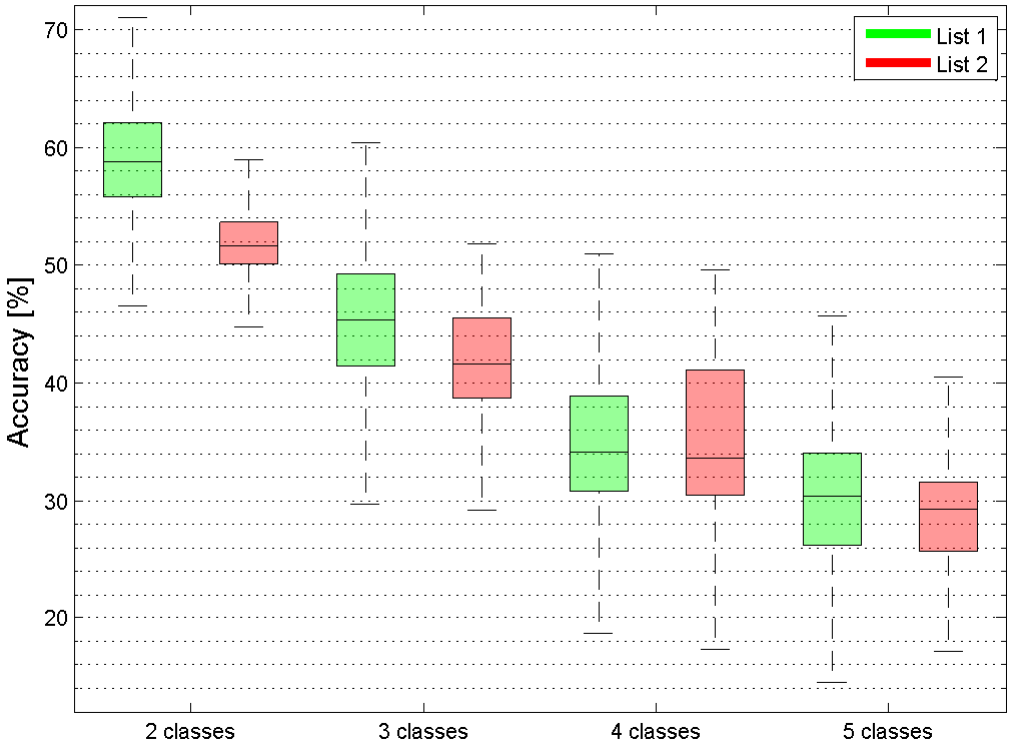


Figure 6.1: Box plot of the distribution of accuracies for each list and number of classes

After looking carefully at this figure it becomes clearer that List 1 is classified better than List 2, regardless of the number of classes, but specially for the case of 2 classes, where the existing gap is maximum. Actually, the best accuracy achieved on List 2 for 2 classes is close to 60%, while on List 1, it has been shown in the previous section that it is possible to obtain accuracies over 70%. We can find an explanation to this difference on the fact that videos retrieved through the relevance type of query contain many more metadata than those obtained with the other two queries, which are the additional videos that define List 2. Therefore, since List 2 is poorer than List 1 in terms of metadata, the cluster analysis becomes more complicated and less accurate to be representative of real polarity differences, which is consequently reflected on the classification results.

For these reasons, in the next layers of analysis, only results of List 1 will be taken into consideration. This way, the analysis will become much simpler and easier to interpret, since it will not

be necessary to maintain the differences between lists in tables and figures.

6.4 Comparison of Metadata Strategies

One important aspect of this research work is the distinction between types of YouTube metadata that led us to define three different clustering strategies in Section 3.5.3, one that uses only explicit-opinion metadata, one that uses implicit-opinion metadata and a third one that combines both types. Thus, one of the most interesting aspects of the evaluation of results is to compare the performance of each strategy to find out if one type of metadata works better than the other or if the combination improves the results. As it has been stated before, this analysis will be only applied on results of List 1.

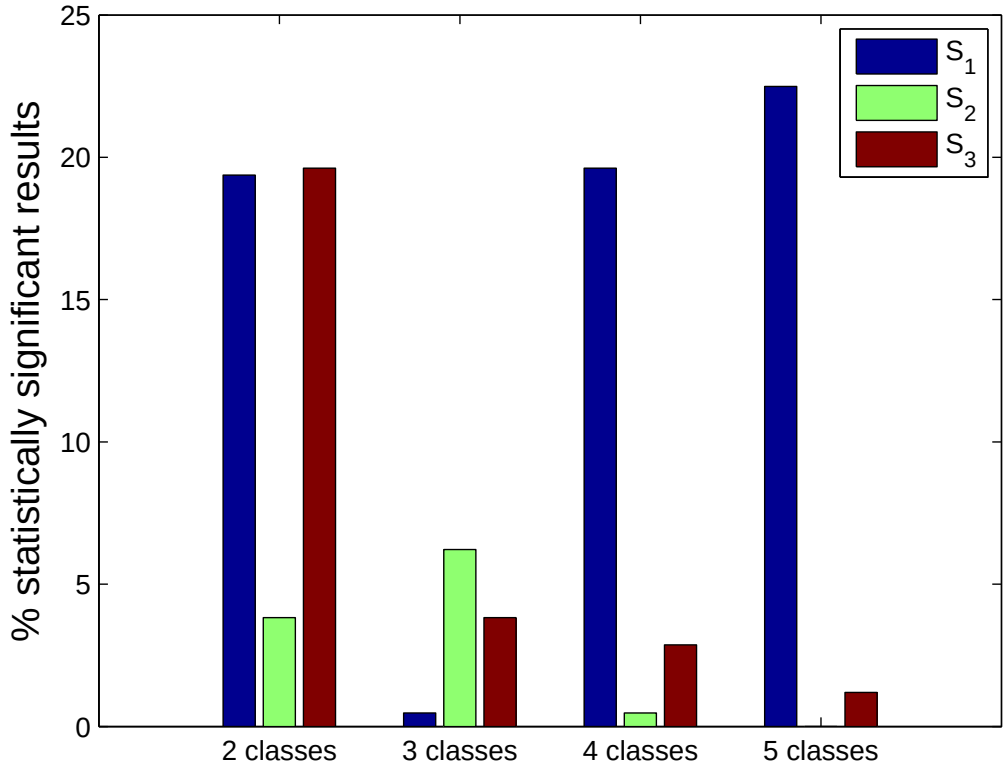


Figure 6.2: Histogram with the percentage of significant results produced by each strategy and number of classes.

The first approach to compare the strategies is to look at the number of statistically significant results that each strategy has produced. In Figure 6.3 the portion of the total number of statistically significant results produced by each strategy has been represented organised by number of classes.

The first conclusion that can be extracted from this figure is that S_2 does not achieve a good performance for any number of classes. It is the strategy with most number of relevant results for 3 classes, but it is still a low number of them, and for 5 classes there are none. Conversely, S_1 is clearly the best strategy for 4 and 5 classes and has approximately the same number of useful results than S_3 for the case of 2 classes. Finally, S_3 proves to work quite well only on 2-classes classification. In order to complete this study, a box plot showing the distributions of percentage of correctly classified instances for every valid experiment (not only relevant results) can be analysed in the next figure. It confirms that S_2 performs significantly worse than the other two strategies and shows that the differences between S_1 and S_3 are not so big as it could be though by looking

at the previous figure, but S_1 performance is still slightly better, especially for 2 classes.

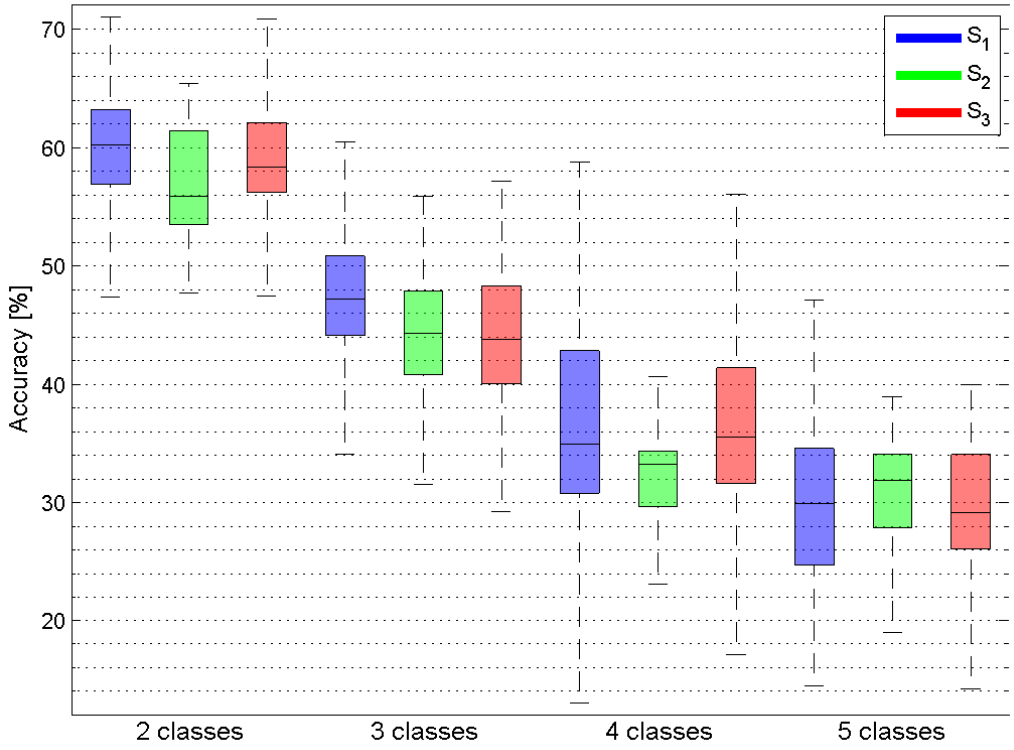


Figure 6.3: Box plot of the distribution of accuracies for each strategy and number of classes.

Explanations to these differences between strategies can be found in the intrinsic nature of the metadata that define each strategy. On the one hand, S_2 has provided the worst results probably because it relies just on one metadatum (viewsCountScore). Having only one feature makes the cluster analysis significantly more difficult and this has important implications on the posterior classification experiments. On the contrary, S_1 relies on 4 different metadata which make the data set richer and the clustering more likely to derive annotations closer to real polarity differences on videos. This is reflected, in turn, on the classification process. Finally, S_3, which has provided similar results to S_1, relies on the same metadata plus the viewsCountScore. It can be thought that the clustering combinations have produced very similar annotations and hence, similar classification results, except in some cases, where the addition of a metadatum of a different nature might have caused incoherent clusters and worse classification results. This phenomenon of diversity in YouTube metadata is explained in detail in [7].

In order to simplify the analysis in the subsequent layer of analysis, only results of S_1, the strategy that uses explicit-opinion metadata, which has proved to provide the most successful and consistent results, are considered.

6.5 Analysis of Features Performance

This level of analysis is perhaps the most interesting one in terms of application of results, since it will allow us to find out which features have been more useful for providing good classification results and thus, are more indicative of the subjective appreciation of videos from users. Since we have to carry out a comparison of 21 features, it would be rather confusing to maintain other levels

of analysis at this point. Therefore, apart from the previously mentioned removal of strategies 2 and 3, only the case of 2-classes classification will be taken into account, which is the case of best performance.

Again, we can illustrate the performance of each feature by plotting the percentage of statistically significant results that have been achieved with the contribution of each feature. Note that, except for those combinations relying on one single feature, most of the presented results were achieved by combining bringing features together.

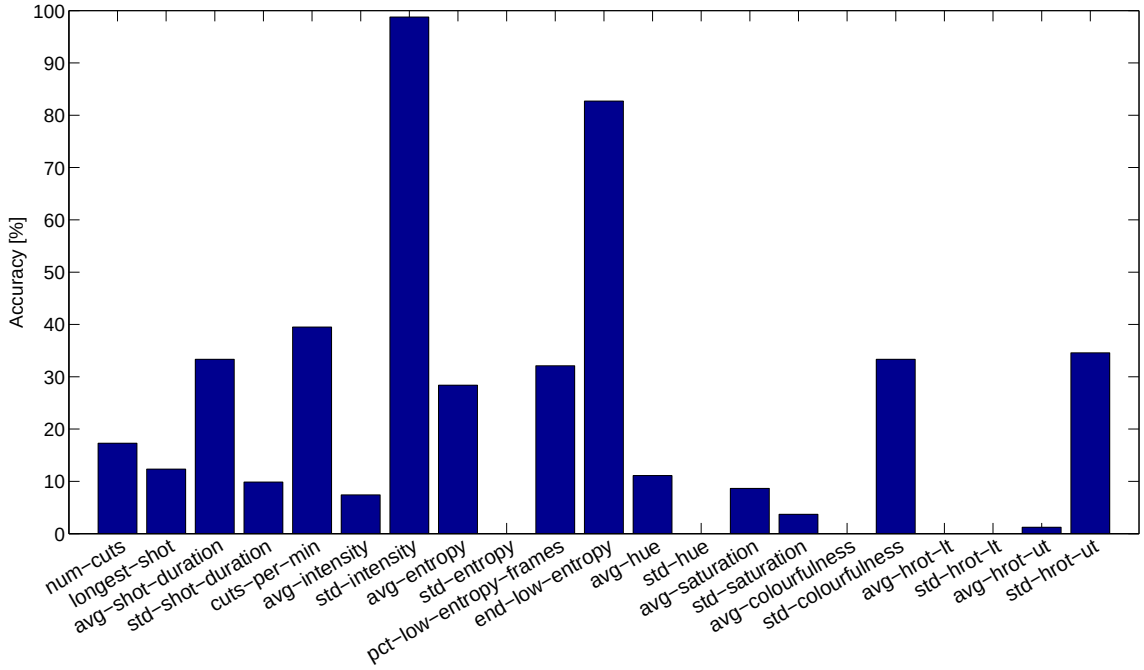


Figure 6.4: Histogram with the percentage of statistically significant results in which each feature was present for S_1 and 2 classes.

In Figure 6.4 we can observe that most successful features regarding the contributions to relevant results are std-intensity, end-low-entropy, cuts-per-min, avg-shot-duration, std-hrot-ut, pct-low-entropy-frames and std-colourfulness. It is important to remark that these features involve all the different visual aspects: intensity, entropy, temporal segmentation, colour and rule of thirds. The average accuracy of the significant results to which these features contributed is around 66–68%.

6.5.1 Comparison with Previous Works

It is quite interesting to compare the features that have proven to better assess the user perception of viewers in our car advertising videos to the ones obtained in previous similar works, for instance, the work of Moorthy *et al.* [32], who evaluated the worth of features assessing aesthetics of consumer videos, and the work and Datta *et al.* [9], who evaluated the aesthetics in pictures. A common feature that has been useful in the three works is the colourfulness, in its different variations. It was proposed in [9] and it was selected in that work as one of the best 15 features. Later, four features related to colourfulness were selected among the best 14 features for assessing consumer videos aesthetics. Here, std-colourfulness has been one of the most useful features. Another common feature is the intensity, which appears repeatedly in the different works as one of the most relevant features. Saturation was also an useful feature for assessing aesthetics of pictures, the same as features describing texture, although the approaches have been different.

Some features that proved to be useful in the previous described aspects such as the frame rate

and the image quality in [32], as they applied their experiments on consumer-generated videos and measures of quality are important. The aspect ratio was also an important feature for assessing picture aesthetics, which seems logical as different formats are used in photography and it can have an impact on the viewer. In our case, we decided not to consider these kind of features for the experimentation because, apart from being interested on other kind of features, they are very low-level features and we have collected videos from the same domain, which share at some extent these characteristics. Interesting is that we have demonstrated that features related to the temporal segmentation, for instance, cuts-per-min and avg-shot-duration, which were not tested by Moorthy *et al.*, are also useful for predicting the appeal of car advertising videos. This has confirmed our intuition exposed in Section 4.2.

An important conclusion is that the correspondence of features among previous works and this one is a prove of consistency of the research on this field, even though the video domains are not identical. This opens lines of research for keep on trying to better model the aesthetics of videos and images using these features with improvements and other new ones.

6.5.2 Best Result of Strategy 1

It is quite interesting to look at the characteristics of the best classification result provided by Strategy 1, the one considered in Section 6.4 to illustrate the best strategy. In the next table we show the most important characteristics of the clustering combination and classification experiment that provided the best accuracy:

Type of metadata	explicit-opinion
Number of classes	2
Accuracy (σ)	70.98 (11.20)
Baseline (σ)	55.05 (2.85)
Clustering distance	correlation
Normalisation	No (only implicit by distance)
Attribute Selection	SVM Attribute Evaluator
Number of features	9
Features	avg-shot-duration, cuts-per-min, std-intensity, avg-entropy, pct-low-entropy-frames, end-low-entropy, avg-hue, std-colourfulness, std-hrot-ut
Classifier	Logistic

Table 6.4: Characteristics of the best classification experiment of Strategy 1

First of all, it is a remarkable fact and good news the high number of features of this experiment. The fact that the best experiment uses 9 features means that many visual descriptors are useful and relevant for modelling the user assessment of a video. Furthermore, it is interesting to contrast the set of features used in this experiment with Figure 6.4. We can observe that these features coincide perfectly with the features that were selected most often in statistically significant experiments. This fact confirms that these are the features that best model the aesthetics and perceptual assessment of a car advertising video.

6.6 Analysis of Classifiers Performance

Although it is not the most important aspect of the analysis of results, it is worth doing a brief comparative analysis of the performance of the different classification algorithms. Again, only results from S_1 and 2 classes are taken into consideration in order to simplify the analysis. Looking at Figure 6.5, it can be observed that three classifiers, BayesNet, SMO and RandomTree did not

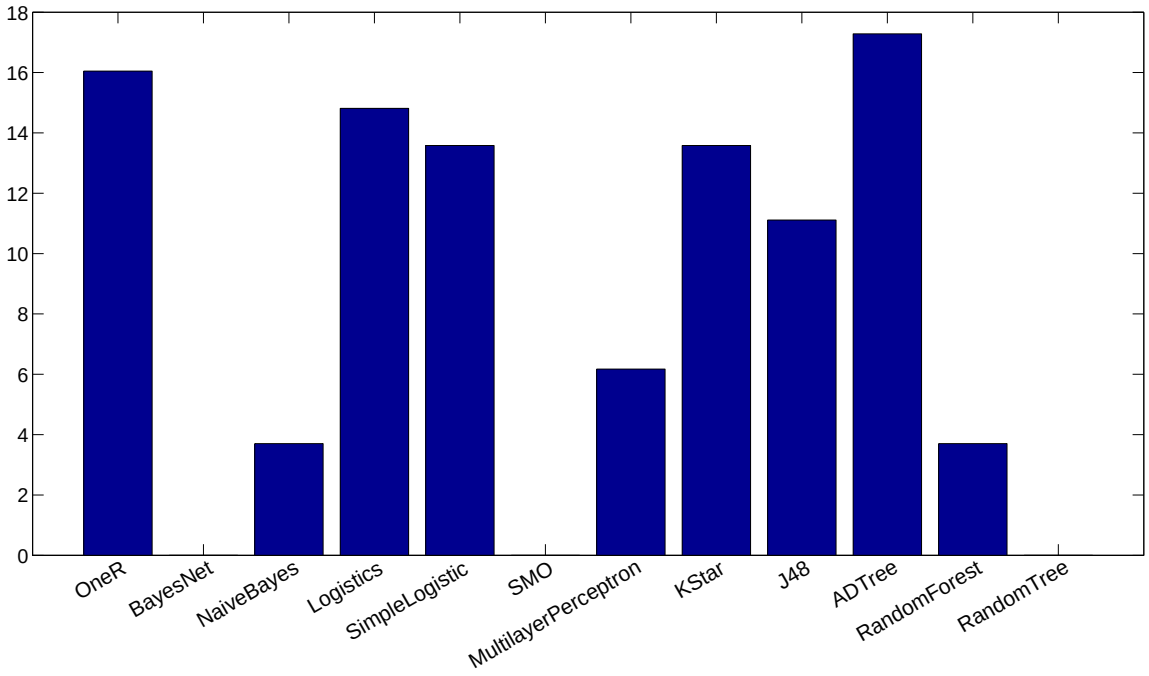


Figure 6.5: Histogram with the percentage of statistically significant results yielded by each classifier for S_1 and 2 classes.

produce any relevant result, whereas there are certain algorithms which performed particularly good. It is the case of OneR, both Logistic classifiers, KStar, J48 and ADTree.

From this evidence, two conclusions can be extracted regarding the performance of different classifiers. First of all, it is evident from the figure that some classifiers have provided more statistically significant results than others, which did not provide any. However, the second conclusion is that there is not any particular classifier which clearly produced more relevant results than others.

Chapter 7

Conclusions and Future Work

The purpose of this chapter is to discuss the most important conclusions of this work and to summarise its main aspects and contributions to the state of the art in the field of assessing the user perception of videos after having analysed the results. The key technical aspects and characteristics that have been used in this project to achieve the objectives and develop the whole process will be also briefly exposed here. Finally, some of the improvements that could be added to this work will be mentioned and justified and we will outline some of the most important and interesting developments and lines of research that could be derived from this work.

7.1 Main Conclusions

In this research work, we have presented a novel approach to address the almost unexplored field of assessing aesthetic quality on videos in terms of user perception, basing our work on an unsupervised learning method to annotate two corpora composed by 138 and 315 professional car advertisements, respectively. We have obtained the polarity labels after performing a cluster analysis of metadata available at YouTube following three different strategies according to the way these metadata are generated by users. Then we have extracted 21 low-level video features and have carried out classification experiments using different feature selection algorithms and classifiers over the three strategies. At the end, we have achieved a maximum classification accuracy of 72.18%, which is very close to the results achieved in the state of art, which have always followed supervised learning approaches . However, as a novel approach to the more challenging problem of annotating the corpus through unsupervised learning methods, we can conclude that we have achieved a successful accuracy.

We believe that this research has made three important contributions.

- First of all, the research on aesthetic quality assessment, which is still in exploration, has been extended by using different methodologies and proposing new video descriptors that we have proven to be useful.
- We have proposed for the first time, to the best of our knowledge, a computational model to assess the user perception of videos without having to rely on ratings provided by users, something which is usually hard to obtain. On the contrary, we rely on metadata freely provided by YouTube and naturally generated by every user who watches a video. We have distinguished two types of metadata and made experiments using them separately and together. It has been shown that explicit-opinion metadata model better the average user perception than automatically generated metadata, such as the number of views.

- Finally, we have extended the research to a number of classes higher than two, and we have obtained statistically significant results over 3, 4 and 5 classes. Previous works in this field had only provided successful results for 2-classes classification.

Along the development of this project, it has been necessary to make use of several different technologies which are worth mentioning at this point in order to better illustrate the process followed to finally extract these conclusions. This technical aspects can be classified into three types:

- In order to acquire a proper corpus on which to make the experimentation, it was necessary to develop Java programmes that interact with YouTube servers through its API. This made possible to automatically query and retrieve the videos and their intrinsic metadata.
- A key part of the research process is the extraction of image and video features which are the base of the experimentation, since one of the main objectives is to find out which features correlate with the user perception of car advertisements. These descriptors have been computed through MATLAB scripts.
- As a work which aims to extract statistical conclusions related to classification, multiple branches of the field of machine learning have been used and developed as a combination of MATLAB scripts and WEKA functionalities:
  - The first classification step relied on the unsupervised learning method of cluster analysis. More specifically, the k-means algorithm was used to derive polarity annotations for every video.
  - The second classification step was supervised learning and consisted in the evaluation of 13 different classifiers implemented on WEKA.
  - An intermediate, but important, step was the feature selection algorithms that allowed reducing the dimensionality and complexity of the data sets.
  - Statistical analysis was applied also on the classification results to extract confidence intervals and relevance.

7.2 Future Work

Several improvements or extensions can be identified in this work, as well as some lines of research that this project has opened. In this section we present the most interesting ones:

- Extend the research to other video domains. Using the knowledge acquired in this work, applied on car advertisements, it would be quite interesting to see how the results apply on different types of videos.
- Enlarge the set of features. Due to time constraints, the set of features tested in this work is limited. Therefore, we would like to perform a deeper analysis by adding more features to the data sets. Some features that could be added are SIFT descriptors, motion detection features or HOG descriptors.
- Multimodal analysis. Apart from video descriptors, one of the most interesting improvements would be to add audio and acoustic features to the experiments. Other works in sentiment analysis have shown that used in combination, audio and video features can increase the classification performance. Furthermore, techniques of natural language processing could be applied as well, for instance to extract useful information from user comments, to improve the cluster analysis, or to analyse the speech in the advertisements.
- Detection of viral videos. An implementation of a system for detecting viral videos would be of great interest for improving the cluster analysis performance.

- Optimisation of a classifier. If we used the results for a particular application, it would be quite interest to develop a procedure for optimising the performance of a particular classifier, which could increase the classification rate.

Chapter 8

Project History

This chapter first includes a summary of the main steps that were taken in the development of the project as a schedule to illustrate the order of the development and the approximate time employed for each task. Additionally, a project budget is included as an estimation of costs derived from the development of this research.

8.1 Schedule

The next schedule shows an approximation of the order of the steps taken during the development of this work and the dates of each task.

1. **Start** (07/11/2013)
   (a) First meeting with the tutor and discussion of general aspects of the project.
   (b) Meeting with Audiovisual Communication professors.
2. **Corpus acquisition** (March 2013 - May 2013)
   (a) Tests with YouTube API (1 week)
   (b) Download of sample code.
   (c) First test queries. (4 days)
   (d) Design of data format for representation of videos metadata.
   (e) Retrieval of basic metadata. (10 days)
   (f) Retrieval of specific and partially *hidden* metadata (likes and dislikes) through explicit parsing. (1 week)
   (g) Retrieval of *hidden* metadata (location and demographic data). (1 week)
   (h) Execution of the programme that automatically downloads all metadata. Tests and re-execution. (10 days)
   (i) Download of videos. (3 days)
3. **Feature extraction** (May 2013 - July 2013)
   (a) Extraction of simplest features: intensity, saturation, hue and entropy. (10 days)
   (b) Extraction of features related to temporal segmentation. (10 days)

(c) Extraction of features related to colourfulness. (15 days)

(d) Extraction of features related to rule of thirds. (15 days)

(e) Implementation of scripts for downloading and merging all features. (1 week)

4. **First clustering and classification experiments** (July 2013 - August 2013)

   (a) Study of the correlations between metadata and their characteristics. (3 days)

   (b) Automatic filtering of the data set. (15 days)

   (c) First manual filtering (by title).

   (d) Construction of the first version of the corpus (>138 instances).

   (e) Clustering of the list of videos in WEKA GUI. (3 days)

   (f) Test first models for classification. (1 week)

   (g) Obtaining first results and conclusions.

5. **Proposal of improvements** (August 2013)

   (a) Extraction of new features: standard deviation of measures, new features related to entropy, new features related to the number of cuts and extension of the features related to the rule of thirds. (10 days)

   (b) Re-filtering: extensive manual filtering by watching videos (138 instances). (5 days)

   (c) New cluster analysis and extensive classification experiments with the WEKA Experimenter. Good results are obtained but a larger data set would be desirable. (1 week)

   (d) Enlargement of the list by using all type of queries (rating, relevance and view count). List 2 with 315 instances is obtained. (10 days)

6. **Design of automatic procedures for testing many combinations** (September 2013 - October 2013)

   (a) Definition of metadata strategies.

   (b) Implementation of automatic clustering in MATLAB. (25 days)

   (c) Implementation of automatic feature selection, classification and analysis of results in MATLAB and using the WEKA Experimenter and Analyser. (1 month)

7. **Evaluation of results** (November 2013)

   (a) Evaluation of results in terms of statistical significance. (5 days)

   (b) Comparison of lists, strategies, classifiers ,etc. (1 week)

   (c) Comparison of features. (5 days)

   (d) Implementation of scripts to plot figures. (15 days)

8. **Writing of articles and project report** (December 2013 - January 2013)

   (a) First version of a paper for journal Elsevier. (20 days)

   (b) First version of the report. (8 weeks)

   (c) Revision of paper and report. (2 weeks)

   (d) Paper for International Conference on Image Processing (ICIP) (10 days)

   (e) Submission of paper for ICIP on February the 14th.

8.2 Budget

The development of this technical project requires certain work and assets that have associated some economic costs. In this section an estimation of all these costs is presented organised into two types of costs: human resources and tangible assets.

8.2.1 Human Resources

Economic costs associated to human resources refer basically to salaries of workers involved in the development of this project. In order to estimate these costs, it is necessary to know the salaries and time invested in the execution of the project. From a total of 12 months of work in the whole development, the author of this project has worked without any salary during 6 months and other 6 months with a 550 €/month fellowship in the Departamento de Teoría de la Señal y Comunicaciones of the Universidad Carlos III de Madrid. Regarding the direction of the project, an estimation of 5 hours of work per week during 50 weeks and a salary of 72 €/hour has been considered.

Name of worker	**Salary**	**Hours**	**Total**
Alejandro Hernández García	7 €/h	600	4,200 €
Fernando Fernández Martínez	72 €/h	250	18,000 €
Total			**22,200 €**

Table 8.1: Summary of the total remunerated hours of work of the author and the tutor.

8.2.2 Tangible Assets

During the realisation of this project a number of different materials and tangible assets have been used and they have an economic cost as well:

- **Laptop**: the personal computer of the author has been used during the whole project, but mainly during the first 6 months of development. It has been used to programme and implement the algorithms, run experiments and store results. It has a cost of 699 €and this price can be depreciated over four years. Therefore, the associated cost to the project can be estimated in 175 €.
- **Computers**: the author and the director have used two computers belonging to the department, each with an associated cost to the project of 100 €.
- **Work area**: the cost of the work area in the department includes electricity, heating, furniture and maintenance, among others. The total cost can be estimated in 1000 €/month. Since it is shared by 6 workers, we can associate a cost of 166 €/month.
- **Software**: most of the software has been chosen to be open-source, i.e. with no cost, except MATLAB. The license of MATLAB R2009b from MathWorks costs 1,950 €, which can be depreciated over four years, yielding a cost of 488 €for this project.
- **Computer cluster**: the most costly tasks have been executed on a computer cluster in the department. It is shared by many users, so we can associate a cost to this project of 200 €.
- **Office materials**: in this section we include the price of photocopies, pens, paper and other materials. The total cost can be estimated in 10 €/month.

Table 8.2 summarises the costs of tangible assets associated to the development of the project.

Item	Price	Quantity	Months	Total
Laptop computer	175 €	1	12	175 €
Desktop computer	100 €	2	6	200 €
Work area	166 €/month	1	6	966 €
MATLAB R2009b	488 €	2	12	976 €
Computer cluster	200 €	1	12	200 €
Office materials	10 €/month	1	6	60 €
Total				**2,577 €**

Table 8.2: Summary of the costs associated to tangible assets.

8.2.3 Final Budget

The total economic cost of the project is computed through the addition of costs related the human resources and costs of tangible assets. The final budget is summarised in Table 8.3

Concept	Price
Human resources	22,200 €
Tangible assets	2,577 €
Subtotal	24,777 €
IVA (21%)	5,203.17 €
Total	**29,980.17** €

Table 8.3: Final budget

Appendices

Appendix A

WEKA
Machine Learning Software

WEKA, which stands for Waikato Environment for Knowledge Analysis, is an open-source software tool that provides functionality of machine learning and was completely developed in Java in the University of Waikato in New Zealand. It is collection of machine learning algorithms that allow performing tasks of classification, clustering, visualisation and feature selection among many others, whose functionality is available both through a graphical user interface and a command-line interpreter. First released in 1997, some of the objectives of the WEKA project, according to its creators, are the following:

- Make machine learning techniques generally available.
- Develop new machine learning algorithms and give them to the world.
- Contribute to a theoretical framework for the field.

WEKA has become a very popular tool for machine learning tasks and is widely used by researchers and industrial scientists. Most of its functionality is gathered in a *wiki*, different manuals and books, but especially in [55].

The graphical interface of WEKA is divided into two parts: the Explorer and the Experimenter. The former is designed for opening a data set and visualising them, for instance through histograms and for applying classification algorithms, clustering or feature selection. The experimenter allows testing different classification algorithms on several data sets on a single run, in order to obtain the overall information regarding the whole process and compare data sets or classifiers. Additionally, it includes an Analyser that performs statistical analysis and organise the results of the Experimenter.

A.1 ARFF

ARFF [53] is the file format used by WEKA algorithms to correctly interpret input data and to store outputs. It stands for Attribute-Relation File Format because it is used to describe a list of instances that share a set of attributes. The development of this format was part of the WEKA project in the University of Waikato. An ARFF file consists of three different parts:

- **File identifier**: a string preceded by the keyword @relation, that gives a title to the file and identifies it. It is defined according to the following format:

```
@relation <relation-name>
```

- **Attributes**: a list with the attributes of the instances. Each attribute is preceded by the keyword @attribute and is given a proper type: numeric, string, nominal or date. The syntax for defining one attribute is the following:

  ```
  @attribute <attribute-name> <datatype>
  ```

- **Body**: the list of instances with their corresponding attribute values, separated by commas, similarly to CSV files. It starts after the keyword @data and each line corresponds to a new instance.

Since we have developed most of the machine learning tasks of this project with WEKA, we decided to represent all our data files according to ARFF specification as .arff files. Therefore, MATLAB scripts were implemented to read and write this type of files and it became the standard data representation format for our project. To better illustrate the format of these kind of files, a portion of the beginning of an .arff file is provided in Figure A.1

```
@relation video_features

@attribute Num_cuts numeric
@attribute Longest_shot numeric
@attribute Mean_shot_duration numeric
@attribute Std_shot_duration numeric
@attribute Mean_cuts_per_min numeric
@attribute Mean_intensity numeric
@attribute Std_intensity numeric
@attribute Mean_saturation numeric
@attribute Std_saturation numeric
@attribute mean_hue numeric
@attribute Std_hue numeric
@attribute Mean_entropy numeric
@attribute Std_entropy numeric
@attribute Pct_low_entropy_frames numeric
@attribute End_low_entropy {1,0}
@attribute Mean_colourfulness numeric
@attribute Std_colourfulness numeric
@attribute Mean_hrot_lt numeric
@attribute Std_hrot_lt numeric
@attribute Mean_hrot_ut numeric
@attribute Std_hrot_ut numeric
@attribute video_id string

@data
15,4,1.3396,0.916,40.9091,100.7936,59.5436,0.2624,0.1338,0.2973,0.1675,6.7384,1.1875,0.0348,0,104.7266,13.6107,0.3833,0.1848,0.3833,0.2062,uB1q82TRagY
14,1.1667,0.62,0.3452,45.5696,42.3919,29.9838,0.1731,0.1256,0.2426,0.1796,4.9695,2.7985,0.3233,1,116.592,23.3483,0.138,0.1241,0.138,0.0977,mKsyHbFPbTE
27,3.6667,0.9131,0.8531,62.3077,100.6973,61.2497,0.2367,0.2746,0.2033,0.1785,5.2464,1.9498,0.1482,1,114.586,21.513,0.1641,0.188,0.1641,0.1313,UtrQ5B_6wBs
51,3.9667,0.6962,0.6914,84.143,88.5217,56.9191,0.4023,0.1477,0.4439,0.1196,6.4978,1.495,0.0183,0,100.6304,20.0696,0.2985,0.1781,0.2985,0.2044,iN1y1AQI6WY
15,6.0667,1.0563,1.3926,52.8376,164.9223,54.7843,0.2082,0.1572,0.2461,0.1715,5.7897,2.4726,0.2495,0,106.0631,28.1631,0.3733,0.242,0.3733,0.3015,yzLKtkYIFoo
15,5.0333,1.7542,1.3466,28.754,60.291,32.7248,0.2119,0.1169,0.3002,0.1407,5.6721,2.0047,0.126,0,103.5192,20.1532,0.2159,0.1321,0.2159,0.1465,FyjKrP4kIgE
39,1.4333,0.5975,0.4495,87.4222,120.0637,57.0112,0.0382,0.1261,0.0411,0.0775,6.2215,1.8253,0.0851,0,117.7367,12.1757,0.2129,0.1758,0.2129,0.1709,KoQ_qxub10E
28,4.5333,1.2437,1.0722,42.8936,57.8268,30.8684,0.219,0.1129,0.3254,0.1384,5.6643,1.8025,0.0955,0,105.302,19.0306,0.19,0.1341,0.19,0.1357,1lJwYLsVfRs
18,2.8667,1.2053,0.8556,42.1875,44.2081,23.1442,0.1573,0.1086,0.1567,0.1115,4.9799,2.5332,0.2148,0,117.3603,20.2779,0.1461,0.1202,0.1461,0.107,LY8r-UvXiOQ
51,3.9667,0.6962,0.6914,84.143,88.5217,56.9191,0.4023,0.1477,0.4439,0.1196,6.4978,1.495,0.0183,0,100.6304,20.0696,0.2985,0.1781,0.2985,0.2044,iN1y1AQI6WY
21,2.7,1.0909,0.6526,47.25,70.3135,46.4182,0.1821,0.1562,0.2371,0.159,6.0231,1.5721,0.0452,0,117.8636,16.7866,0.1784,0.1781,0.1784,0.1533,H6OfL1JV46Y
10,3.8333,2.0939,1.1227,22.8717,81.4326,43.7099,0.254,0.1628,0.361,0.1651,6.2407,2.6283,0.1613,0,103.7345,24.4736,0.3065,0.2339,0.3065,0.1981,9b4JXuC_hx8
15,2.5333,1.2333,0.6205,39.9408,138.1028,40.5261,0.1206,0.0985,0.3601,0.1038,7.0404,1.4081,0.0415,0,103.1835,14.0595,0.2054,0.1261,0.2054,0.1067,LSL147vLc8s
```

Figure A.1: Portion of the beginning of an ARFF file

Appendix B

Pseudo-code of Relevant Algorithms

The aim of this appendix is to offer the pseudo-code of some of the most relevant algorithms that have been implemented along this research project for carrying out the required tasks, such as retrieval of metadata through the YouTube API, filtering, extraction of features or procedures for executing machine learning tasks in WEKA or MATLAB. With the summary of these algorithms it is intended to better illustrate every important procedure of this project.

B.1 Corpus Acquisition

This section gathers the algorithms related to the videos acquisition process, detailed in Chapter 3.

```
Data: Queries Files with Query Lines
Result: a file for each video containing all its metadata
Connection to YouTube server;
foreach querytype ∈ {relevance,rating,viewCount} do
    Read query file;
    while readLine ≠ null do
        Read line;
        Parse line and extract search parameters;
        Raise a YouTube query with these parameters;
        Obtain VideoFeed: set of videos returned by query;
        foreach video v ∈ VideoFeed do
            Extract accesible metadata;
            Run XML parser to extract number of likes and dislikes;
            Run HTTP response parser to extract insight data;
            Retrieve CommentFeed from v;
            Write metadata into specific metadata file for v;
            Write comments into specific comments file for v;
            Write videoID, URL and title into summary file;
        end
    end
end
Close connection with YouTube server;
```

Algorithm 1: Summary of the algorithm for querying the videos and retrieving their metadata

Algorithm 1 is a summary of the algorithm implemented in Java for communicating with the YouTube server through its API and raising the queries that allow retrieving the metadata of the videos that make up the data sets. It is implemented so that the retrieved videos satisfy certain requirements specified in TXT files, which are the input for the algorithm. Algorithm 2 shows the steps of the filtering process that allows removing from the lists those videos not satisfying the requirements presented in Section 3.2.

```
Data: Metadata Files
Result: Filtered Data Sets
foreach list ∈ {List 1,List 2} do
    Read metadata files of list;
    foreach video v ∈ list do
        % Delete repeated instances
        if v.videoID appears more than once then
            Keep only one instance with v.videoID;
        end
        % Duration filter
        if v.duration < 10 | v.duration > 115 then
            Delete instance v;
        end
        % Metadata filter
        if v.numRaters < 3 then
            Delete instance v;
        end
        % Key words filter
        if v.title contains ("spot" OR "campaign" OR "publi*" OR "advertisement") then
            Keep instance v;
        else
            if v.description contains ("spot" OR "campaign" OR "publi*" OR
             "advertisement") then
                Keep instance v;
            else
                Delete instance v;
            end
        end
    end
    Re-write filtered list;
end
```

Algorithm 2: Summary of the algorithm for automatically filtering the data sets

B.2 Feature Extraction

In this section we present summaries of the algorithms that have been implemented for extracting the image and video features explained in Chapter 4.

Algorithm 3 shows a summary of the operations required for computing the feature values of the descriptors related to the frame intensity. A similar procedure is followed for computing the features related to hue, saturation and some of the entropy.

Data: JPEG files of frames; N: number of frames, W: width, H: height
Result: Colourfulness features
for $n = 1$ **to** N **do**
  $frame = frames(n)$;
  $frameIntensity(n) = 0$;
  for $i = 0$ **to** W **do**
    for $j = 0$ **to** H **do**
      $frameIntensity(n) = frameIntensity(n) + frame(i, j)$;
    end
  end
  if $frameIntensity(n) = 0$ **then**
    do not consider this frame and update N conveniently ;
  end
  $frameIntensity(n) = \frac{frameIntensity(n)}{H \times W}$;
end
$avgIntensity = \frac{1}{N} \sum_{n=1}^{N} frameIntensity(n)$;
$stdIntensity = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (frameIntensity(n) - avgIntensity)^2}$;

Algorithm 3: Summary of the algorithm for extracting intensity related features

Algorithm 4 summarises the procedures for computing the average colourfulness feature and the standard deviation of the distribution along all the frames of a video.

Data: JPEG files of frames; N: number of frames, W: width, H: height
Result: Colourfulness features
Obtain the four equally-spaced regions according to the range of the Lab space ;
`% Define the ideally multi-coloured distribution`
$ideal = \frac{1}{64} \times ones(1, 64)$;
for $n = 1$ **to** N **do**
  $frame = frames(n)$;
  Convert frame to the *Lab* colour space ;
  `% Compute colour histogram of frame`
  $colourHist = zeros(4, 4, 4)$;
  for $r = 1$ **to** 4 **do**
    for $g = 1$ **to** 4 **do**
      for $b = 1$ **to** 4 **do**
        Count number pixels in $frame$ with a value in block (r, g, b) ;
        $colourHist(r, g, b) =$ #pixels-in-this-block ;
      end
    end
  end
  Normalise the array by the total number of pixels ;
  `% Get the value of colourfulness by computing the EMD`
  if $frameIntensity(n) = 0$ **then**
    do not consider this frame and update N conveniently ;
  end
  $frameColourfulness(n) = EMD(ideal, colourHist)$;
end
$avgColourfulness = \frac{1}{N} \sum_{n=1}^{N} frameColourfulness(n)$;
$stdColourfulness = \sqrt{\frac{1}{N} \sum_{n=1}^{N} (frameColourfulness(n) - avgColourfulness)^2}$;

Algorithm 4: Summary of the algorithm for extracting colourfulness related features

The procedures for extracting the feature values related to the rule of thirds are summarised in Algorithm 5. This is a feature specifically designed for this research work. Unlike other algorithms presented in this section, this one shows only the operations for computing the measure of utilisation of the rule of thirds in a single frame. The average and standard deviation of this feature of the whole video is computed using the values of the measure of every frame.

Data: JPEG picture; W: width, H: height
Result: Measures of degree of utilisation of the rule of thirds for the upper and the lower third lines

$rowUpperThird = \frac{1}{3}H$;
$rowLowerThird = \frac{2}{3}H$;
$rowMiddle = \frac{1}{2}H$;

`% Get sub-frames yielded by the upper third line (UTL)`
$UTLtopSubframe = frame(1 : rowUpperThird, :, :)$;
$UTLbottomSubframe = frame(rowUpperThird : H, :, :)$;

`% Get sub-frames yielded by the lower third line (LTL)`
$LTLtopSubframe = frame(1 : rowLowerThird, :, :)$;
$LTLbottomSubframe = frame(rowLowerThird : H, :, :)$;

`% Get sub-frames yielded by the middle line (ML)`
$MLtopSubframe = frame(1 : rowMiddle, :, :)$;
$MLbottomSubframe = frame(rowMiddle : H, :, :)$;

`% Compute colour histograms of all sub-frames`
$histUTLtop = colourHist(UTLtopSubframe)$;
$histUTLbottom = colourHist(UTLbottomSubframe)$;
$histLTLtop = colourHist(LTLtopSubframe)$;
$histLTLbottom = colourHist(LTLbottomSubframe)$;
$histMTLtop = colourHist(MLtopSubframe)$;
$histMTLbottom = colourHist(MLbottomSubframe)$;

`% Compute initial measure of degree of utilisation of the rule of thirds for the upper third line`
$DiffUTL = 32 \cdot \frac{1}{64 \cdot H \cdot W} \sum_{b=1}^{64} |histUTLtop(b) - histUTLbottom(b)|$;
`% Compute initial measure of degree of utilisation of the rule of thirds for the lower third line`
$DiffLTL = 32 \cdot \frac{1}{64 \cdot H \cdot W} \sum_{b=1}^{64} |histLTLtop(b) - histLTLbottom(b)|$;
`% Compute difference between the two halves of the frame`
$DiffML = 32 \cdot \frac{1}{64 \cdot H \cdot W} \sum_{b=1}^{64} |histMLtop(b) - histMLbottom(b)|$;

`% Apply penalties`
if $(\frac{DiffUTL}{DiffML} \leq 0.95 \ AND \ \frac{DiffLTL}{DiffML} \leq 0.95)$ **then**
  $DiffUTL = 0.33 \cdot DiffUTL$;
  $DiffLTL = 0.33 \cdot DiffLTL$;
else
  if $(DiffUTL > DiffLTL \ AND \ DiffLTL > 0.7)$ **then**
    $DiffLTL = 0.5 \cdot DiffLTL$;
  end
  if $(DiffLTL > DiffUTL \ AND \ DiffUTL > 0.7)$ **then**
    $DiffUTL = 0.5 \cdot DiffUTL$;
  end
end

Algorithm 5: Summary of the algorithm for computing the measures of degree of utilisation of the rule of thirds on the upper and the lower third lines for a single frame

B.3 Classification

This sections compiles the pseudo-code of algorithms concerning the procedures required for performing the operations explained in Chapter 5, namely feature selection, classification with the WEKA Experimenter and statistical analysis. Algorithm 6 shows the procedures for performing the feature selection, Algorithm 7 illustrates the actions taken by the script that executes the WEKA Experimenter and Algorithm 8 the ones of the Analyser.

Data: Data set with 21 feature values for each instance and a class identifier
Result: 44 feature subsets with a smaller number of features
`% CfsSubsetEval / BestFirst`
Define input path and output path according to the data set ;
Define evaluator and search method lines with proper parameters of configuration ;
Execute shell script with arguments: input, output, evaluator and search method ;

`% SVMAttributeEval / Ranker`
for $N = 1$ **to** 10 **do**
  Define input path and output path according to the data set and N ;
  Define evaluator and search method lines with proper parameters of configuration, including N ;
  Execute shell script with arguments: input, output, evaluator and search method ;
end

`% ConsistencySubsetEval / GreedyStepwise`
for $N = 1$ **to** 10 **do**
  Define input path and output path according to the data set and N ;
  Define evaluator and search method lines with proper parameters of configuration, including N ;
  Execute shell script with arguments: input, output, evaluator and search method ;
end

`% InfoGainAttributeEval / Ranker`
for $N = 1$ **to** 10 **do**
  Define input path and output path according to the data set and N ;
  Define evaluator and search method lines with proper parameters of configuration, including N ;
  Execute shell script with arguments: input, output, evaluator and search method ;
end

`% PrincipalComponents / Ranker`
for $N = 1$ **to** 12 **do**
  Define input path and output path according to the data set and N ;
  Define evaluator and search method lines with proper parameters of configuration, including N ;
  Execute shell script with arguments: input, output, evaluator and search method ;
end

`% ClassifierSubsetEval / RaceSearch`
Define input path and output path according to the data set ;
Define evaluator and search method lines with proper parameters of configuration ;
Execute shell script with arguments: input, output, evaluator and search method ;

Algorithm 6: Summary of module Attribute Selection

Data: Specification of one data set and the references to its 44 feature subsets
Result: 44 ARFF files with the classification results of each feature subset
for $subset = 1$ **to** 44 **do**
  Build path of feature subset *subset* (input) ;
  Build configuration by setting output path on a template XML config-file ;
  Execute shell script with arguments: input and configuration file ;
end

Algorithm 7: Summary of module Experimenter

Data: Specification of one data set and the references to its 44 experimenter files
Result: 44 CSV and TXT files with the average accuracy, standard deviation and significance of the classifiers
for $experimenterFile = 1$ **to** 44 **do**
  Build path of experimenter $experimenterFile$ (input) ;
  Execute shell script to generate CSV file ;
  Execute shell script to generate TXT file ;
end

Algorithm 8: Summary of module Analyser

Bibliography

[1] Gediminas Adomavicius and Alexander Tuzhilin. Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, Jun 2005.

[2] Subhabrata Bhattacharya, Behnaz Nojavanasghari, Dong Liu, Tao Chen, Shih-Fu Chang, and Mubarak Shah. Towards a comprehensive computational model for aesthetic assessment of videos. In *ACM Multimedia*, Grand Challenge, October 2013.

[3] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *ARTIFICIAL INTELLIGENCE*, 97:245–271, 1997.

[4] David Bordwell and Kristin Thompson. *El arte cinematográfico: una introducción*. Paidós Comunicación 68 Cine, 4 edition, 1995.

[5] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] Darin Brezeale and Diane J. Cook. Automatic video classification: A survey of the literature. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, (3):416–430, Oct 2008.

[7] G. Chatzopoulou, Cheng Sheng, and Michalis Faloutsos. A first step towards understanding popularity in youtube. In *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, pages 1–6, 2010.

[8] John G. Cleary and Leonard E. Trigg. K*: An instance-based learner using an entropic distance measure. In *12th International Conference on Machine Learning*, pages 108–114, 1995.

[9] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Studying aesthetics in photographic images using a computational approach. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part III*, ECCV'06, pages 288–301, Berlin, Heidelberg, 2006. Springer-Verlag.

[10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[11] Allan C. Elliot and Wayne A. Woodward. *Statistical Analysis: Quick Reference Guidebook*. Sage Publications, Inc., 1 edition, 2007.

[12] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proceeding of the Sixteenth International Conference on Machine Learning*, pages 124–133, Bled, Slovenia, 1999.

[13] M. A. Hall. *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand, 1998.

[14] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, nov 2009.

[15] Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

[16] HDW. Source of figure 4.3a. `http://static.hdw.eweb4.com/media/thumbs/1/88/876621.jpg`, June 2013.

[17] JSOUP. Jsoup parser. `http://jsoup.org`, June 2013.

[18] V. Kathiresan and P. Sumathi. An efficient clustering algorithm based on z-score ranking method. In *Computer Communication and Informatics (ICCCI), 2012 International Conference on*, pages 1–4, 2012.

[19] Yan Ke, Xiaoou Tang, and Feng Jing. The design of high-level features for photo quality assessment. In *CVPR (1)*, pages 419–426. IEEE Computer Society, 2006.

[20] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation*, 13(3):637–649, 2001.

[21] Mohamed A. Khamsi and William A. Kirk. *An Introduction to Metric Spaces and Fixed Point Theory*. Wiley, 2001.

[22] Shehroz S. Khan and Daniel Vogel. Evaluating visual aesthetics in photographic portraiture. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, CAe '12, pages 55–62, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[23] Irena Koprinska and Sergio Carrato. Temporal video segmentation: A survey. In *Signal Processing: Image Communication*, pages 477–500, 2001.

[24] Niels Landwehr, Mark Hall, and Eibe Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

[25] S. le Cessie and J.C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

[26] S. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 2006.

[27] Wei Luo, Xiaogang Wang, and Xiaoou Tang. Content-based photo quality assessment. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *ICCV*, pages 2206–2213. IEEE, 2011.

[28] Yiwen Luo and Xiaoou Tang. Photo and video quality evaluation: Focusing on the subject. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 386–399, Berlin, Heidelberg, 2008. Springer-Verlag.

[29] Luca Marchesotti, Florent Perronnin, Diane Larlus, and Gabriela Csurka. Assessing the aesthetic quality of photographs using generic image descriptors. In *ICCV*, pages 1784–1791, 2011.

[30] MathWorks. Converting color data between color spaces. `http://www.mathworks.es/es/help/images/converting-color-data-between-color-spaces.html`, June 2013.

[31] MathWorks. k-means clustering. Technical report, MathWorks, September 2013.

[32] Anush K. Moorthy, Pere Obrador, and Nuria Oliver. Towards computational models of the visual aesthetic appeal of consumer videos. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 1–14, Berlin, Heidelberg, 2010. Springer-Verlag.

[33] Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *International Conference on Multimodal Interfaces (ICMI 2011)*, Alicante, Spain, Nov 2011.

[34] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: capturing favorability using natural language processing. In John H. Gennari, Bruce W. Porter, and Yolanda Gil, editors, *K-CAP*, pages 70–77. ACM, 2003.

[35] Yuzhen Niu and Feng Liu. What makes a professional video? a computational aesthetics approach. *IEEE Trans. Circuits Syst. Video Techn.*, 22(7):1037–1049, 2012.

[36] International Commission on Illumination. *Colorimetry: technical report*. CIE technical report. Commission internationale de l'Eclairage, CIE Central Bureau, 2004.

[37] Michael Ondaatje and Walter Murch. *El arte del montaje*. Plot Ediciones, 1 edition, 2007.

[38] Oracle. Oracle binary code license agreement for the java se platform products and javafx. `http://www.oracle.com/technetwork/java/javase/terms/license/index.html`, April 2013.

[39] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *In proceedings of EMNLP*, pages 79–86, 2002.

[40] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[41] Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[42] Quizlet. Source of figure 4.5b. `http://o.quizlet.com/i/bNu8gTyuh2sNU4-RtNC20w_m.jpg`, June 2013.

[43] Flickr (Trey Ratcliff). Source of figure 4.4. `http://www.flickr.com/photos/stuckincustoms/1928559400/`, June 2013.

[44] Flickr (Marc Roberts). Source of figure 4.6. `http://www.flickr.com/photos/marcroberts/6198696988/`, June 2013.

[45] Veronica Perez Rosas, Rada Mihalcea, and Louis-Philippe Morency. Multimodal sentiment analysis of spanish online videos. *IEEE Intelligent Systems*, 28(3):38–45, 2013.

[46] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 59–, Washington, DC, USA, 1998. IEEE Computer Society.

[47] Andreas E. Savakis, Stephen P. Etz, and Alexander C. P. Loui. Evaluation of image appeal in consumer photography. volume 3959, pages 111–120, 2000.

[48] Alvy Ray Smith. Color gamut transform pairs. *SIGGRAPH Comput. Graph.*, 12(3):12–19, August 1978.

[49] Marc Sumner, Eibe Frank, and Mark Hall. Speeding up logistic model tree induction. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 675–683. Springer, 2005.

[50] TDYLF. Infographic: Colorizing walter whites decay. `http://tdylf.com/2013/08/11/infographic-colorizing-walter-whites-decay`, December 2013.

[51] PHOTO Technique. Source of figure 4.5a. `http://www.phototechnique.com/how-to/understanding-the-rule-of-thirds/`, June 2013.

[52] TuDesguace.com). List of car brands currently sold in spain. `http://www.tudesguace.com/marcas-coches.html`, June 2013.

[53] New Zealand University of Waikato. Attribute-relation file format (arff). http://www.cs.waikato.ac.nz/~ml/weka/arff.html, July 2013.

[54] Adam Westerski. Sentiment analysis: Introduction and the state of the art overview. Technical report, Jun 2009.

[55] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[56] Martin Wollmer, Felix Weninger, Tobias Knaup, Bjorn Schuller, Congkai Sun, Kenji Sagae, and Louis-Philippe Morency. Youtube movie reviews: Sentiment analysis in an audio-visual context. *IEEE Intelligent Systems*, 28(3):46–53, 2013.

[57] Chun-Yu Yang, Hsin-Ho Yeh, and Chu-Song Chen. Video aesthetic quality assessment by combining semantically independent and dependent features. In *ICASSP*, pages 1165–1168. IEEE, 2011.

[58] Boon-Lock Yeo and Bede Liu. Rapid scene analysis on compressed video. *IEEE Trans. Cir. and Sys. for Video Technol.*, 5(6):533–544, December 1995.

[59] YouTube. Five stars dominate ratings. http://youtube-global.blogspot.com.es/2009/09/five-stars-dominate-ratings.html, December 2013.

[60] YouTube. Google data apis client library. https://developers.google.com/gdata/javadoc/, April 2013.

[61] YouTube. Terms of service of youtube api. https://developers.google.com/youtube/terms, March 2013.

[62] YouTube. The video page gets a makeover. http://youtube-global.blogspot.com.es/2010/01/video-page-gets-makeover.html, December 2013.

[63] YouTube. Youtube api v2.0 - ratings. https://developers.google.com/youtube/2.0/developers_guide_protocol_ratings, December 2013.

[64] YouTube. Youtube statistics. http://www.youtube.com/yt/press/statistics.html, November 2013.

[65] youtube dl. youtube-dl. http://rg3.github.io/youtube-dl/, July 2013.